Implementation Of Low Power Viterbi Decoder On FPGA

A thesis report submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

Under the guidance of:

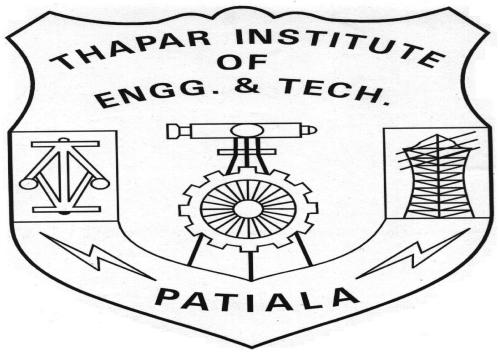
Sh. Sanjay Sharma

Asstt. Prof., E. C. E. D.

Submitted by:

Pushpinder Kaur

Roll No. 8044117



THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY

(DEEMED UNIVERSITY)

PATIALA – 147004

JUNE, 2006

ACKNOWLEDGEMENT

To discover, analyze and to present something new is to venture on an untrodden path towards an unexplored destination is an arduous adventure unless one gets a true torchbearer to show the way. This enlightening guidance, I found in my revered guide Mr. Sanjay Sharma, Assistant Professor, Electronics & Communication Engineering Department, Thapar Institute of Engineering & Technology (Deemed University), Patiala, without whose patronization it was never possible to give final shape to this thesis. I express my heartfelt gratitude towards him for his valuable guidance, encouragement, constant involvement, inspiration and the enthusiasm with which he solved my difficulties.

I shall be failing in my duties if I do not express my deep sense of gratitude towards Dr. R. S. Kaler, Professor & Head of the Department, Electronics & Communication Engineering Department and Dr. A.K. Chatterjee, P.G. Coordinator, Electronics and Communication Engineering Department.

I would also like to thank all the staff members and my co-students who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

I am also thankful to the authors whose works I have consulted and quoted in this work. Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

(Pushpinder Kaur)

TABLE OF CONTENTS

CONTENTS **PAGE NO.**

CONTENTS **PAGE NO.**

CONTENTS **PAGE NO.**

LIST OF FIGURES

Figure Title **Page No.**

LIST OF TABLES

TABLE TITLE **PAGE NO.**

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
FPGA	Field Programmable Gate Arrays
HDL	Hardware Description Language
JTAG	Joint Test Action Group
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
LUT	Look Up Table
MSB	Most Significant Bit
NASA	National Aeronautics and Space Administration
NCD	Native Circuit Description File
NGD	Native Generic Description File
PROM	Programmable Read Only Memory
RE	Register Exchange
SNR	Signal to noise ratio
TB	Trace Back
VA	Viterbi Algorithm
VD	Viterbi Decoder
VHDL	Very High Speed Hardware Description Language
VHSIC	Very High speed Integrated Circuit

ABSTRACT

Convolutional encoding is a forward error correction technique that is used for correction of errors at the receiver end. The two decoding algorithms used for decoding the convolutional codes are Viterbi algorithm and Sequential algorithm. Sequential decoding has advantage that it can perform very well with long constraint length. convolutional codes, but it has a variable decoding time. Viterbi decoding is the best technique for decoding the convolutional codes but it is limited to smaller constraint lengths. The basic building blocks of Viterbi decoder are branch metric unit, add compare and select unit and survivor memory management unit. The two techniques for decoding the data are traceback (TB) method and Register Exchange (RE) method.

TB method is used for longer constraint lengths but is has larger decoding time. Also extra circuitry is required to reverse the decoded bits. The RE method is simpler and faster than the TB method for implementing the VD. RE method is not appropriate for decoders with long constraint lengths.

In this thesis, Viterbi decoder with modified register exchange is implemented. Its specifications are the coding rates 1/3, with the generator polynomial of 171, 165 and 133 respectively and constraint length is 7 with 128 states .The bit serial architecture combined with the modified register exchange method are proposed which reduces the power dissipation. Pointer concept is used for implementing the survivor memory unit of the VD. A pointer is assigned to each register or memory location. The trace-back operation is eliminated in the new architecture, and the amount of memory is reduced. The new Viterbi decoder has efficient memory organization, low hardware complexity and lower power dissipation.

Viterbi decoder is implemented on FPGA. FPGAs reprogrammability and high degree of parallelism attracts them for DSP applications.

The hardware description language VHDL is used to describe the design. The design is synthesized using Xilinx Project Navigator software and simulated using Model Sim. The design implementation is done on Xilinx Spartan 2E xc2s15-6cs144.

CHAPTER-1
INTRODUCTION

1.1 ERROR CONTROL FOR DATA COMMUNICATION

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.

1.2 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 are channel encoder, binary modulator, channel, demodulator, detector and channel decoder.

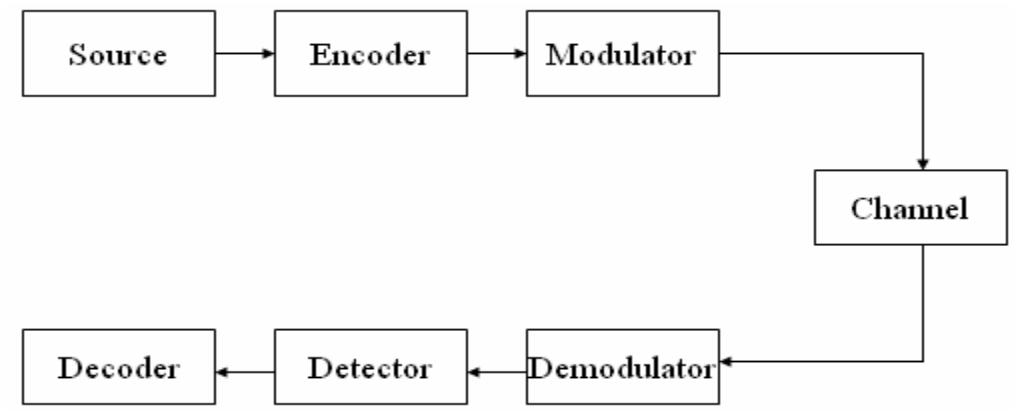


Figure 1.1: Digital Communication System

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 called as hard decision. If quantization is used for the decision process, it is called 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].

1.3 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].

1.3.1 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].

1.3.2 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 [5].

Convolutional code definition parameters are the following: code rate (R), generating polynomial g (n), and 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 1/2 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 than 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 modulo 2 to each other and form output signal [6].

There are three alternative methods that are often used to describe the convolutional code. These are the tree diagram, state diagram and trellis diagram. There exist 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 [7]. 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].

1.4 MOTIVATIONS AND OBJECTIVE

Recently, convolutional codes have become more and more important in digital transmission. A convolutional code with Viterbi decoding is used in wireless communication and satellite communication. The various examples are cellular phone i.e. GSM, IS-54 digital cellular phone standards and IS-95 CDMA standard, modems and video and audio broadcasting. Motivation for low power has been derived from needs to increase the speed, to extend the battery life and to reduce the cost

1.5 LITERATURE SURVEY

The pioneering work on coding and coded waveform for digital communication was done by Shannon (1948), Hamming (1950) and Golay (1949). These works were followed with papers on code performance by Gilbert (1952), new codes by Muller (1954) and Reed (1954), and coding techniques for noisy channels by Elias (1954,1955) and Slepian (1956). The major development was invention of convolutional codes by Elias (1955). The major problem in convolutional coding was decoding. Wozencraft and Reiffen (1961) described a sequential decoding algorithm for convolutional codes. This algorithm was later modified and refined by Fano (1963), and it is now called Fano algorithm. Subsequently, stack algorithm was devised by Ziganzirov (1966) and Jelinek (1969), and Viterbi algorithm was devised by Viterbi (1967). The optimality and the relatively modest complexity for small constraint lengths have served to make the Viterbi algorithm the most popular in decoding of convolutional codes with constraint length less than 10.

Forney (1973-74) showed that the Viterbi algorithm provides both a maximum-likelihood and a maximum posterior (MAP) decoding algorithm for convolutional codes. One of the most important contributions in coding during 1970s was the work of Ungerboeck and Csajka (1976) on coding for bandwidth-constrained channels. In this paper, it was demonstrated that a significant coding gain can be achieved through the introduction of redundancy in a bandwidth-constrained channel and trellis codes were described for achieving coding gains 0f 3-4 db.

Peter Bonek, Andre Ivanov and Samir Kallel (1993) proposed variable rate Viterbi decoder for decoding convolutional codes ranging from rate 7/8 to 1/4 derived from same code rate 1/2. Bupesh Pandita and Subir K Roy (1998) described the design and implementation of Viterbi decoder using FPGA for constraint length 9. The issues related to organization of path memory, decision memory reading techniques and clocking mechanism were discussed. Kang (1998) presented low power Viterbi decoder for CDMA mobile terminals. Erik Paaske and Jakob Dahl Andersen (1998) proposed high speed Viterbi decoder architecture.

M.Kivioja, J.Isoaho and L.Vanska (1999) presented the implementation of Viterbi algorithm on FPGA. Speed performance, easy routability and minimization of inter chip

connections were main design criteria. Yun-Nan Chang, Hiroshi Suzuki, and Keshab K. Parhi (2000) proposed low power bit serial Viterbi decoder architecture.

John Davis, Andrew Lin, Njuguna Njoroge, Ayodele Thomas (2002) proposed Viterbi algorithm for streamline applications. YOU yu-xin, WANG Jin-xiang, LAI Feng-chang and YE Yi-zheng (2002) proposed VLSI design and implementation of high speed Viterbi decoder. Low power dissipation, modified T-algorithm and modified trace back methods were main considerations.

Dalia A. El-Dib and M. I. Elmasry (2005) proposed memoryless Viterbi decoder. This implements the pointer concept with modified register exchange method and bit serial architecture.

1.6 ORGANIZATION OF THESIS

The thesis opens with the basic concepts of channel coding. The first chapter deals with introduction of error correction coding in digital communication systems, channel coding, and types of coding, motivation and objective and literature survey. In the second chapter, convolutional codes, Viterbi algorithm and Viterbi decoder are discussed. Third chapter deals with low power Viterbi decoder architecture and applications of Viterbi decoder and fourth chapter is about field programmable gate arrays and VHDL. Fifth chapter gives the simulation and synthesis results of Viterbi decoder. The last chapter deals with conclusion and future scope.

CHAPTER-2

CONVOLUTIONAL CODES AND VITERBI DECODING

2.1 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 below illustrates a simple convolutional coder with k=1, K=3, n=3, g_1 (n) = (1 0 1), g_2 (n) = (1 1 1), g_3 (n) = (0 1 1) and R=1/2.

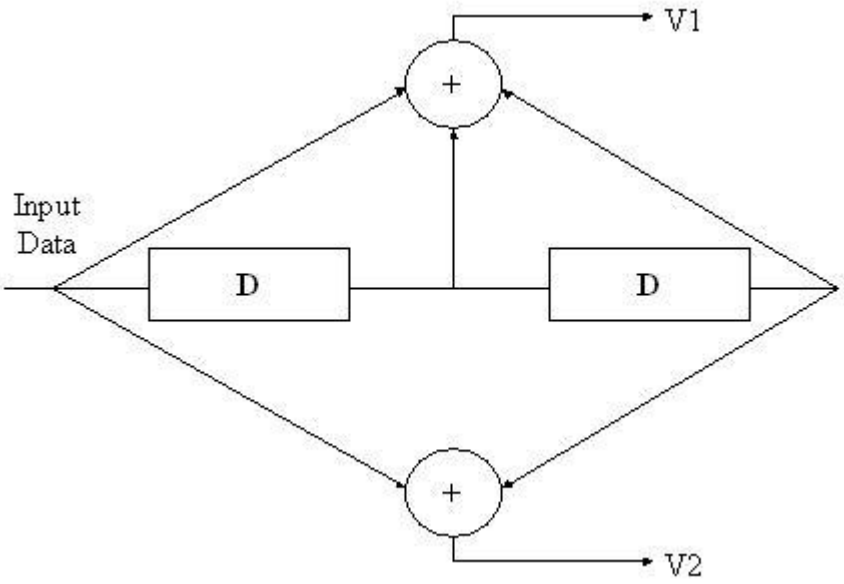


Figure 2.1: Convolutional Encoder

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 output symbol can be described as a function of input symbol and the state. State diagram shows the transition between different states. 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 [3].

Input u	Present state (S_1,S_0)	Next state (S_1,S_0)	Output (v_1,v_2)
0	0 0	0 0	0 0
1	0 0	0 1	1 1
0	0 1	1 0	1 1
1	0 1	1 1	0 0
0	1 0	0 0	1 0
1	1 0	0 1	0 1
0	1 1	1 0	0 1
1	1 1	1 1	1 0

Figure 2.2: State Table

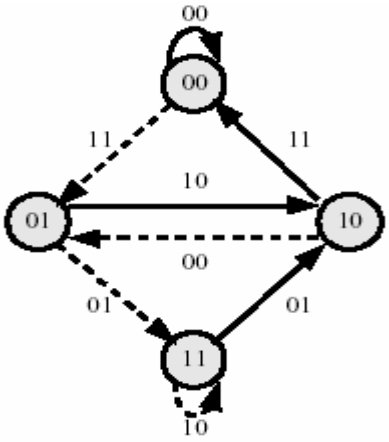


Figure 2.3: State Diagram

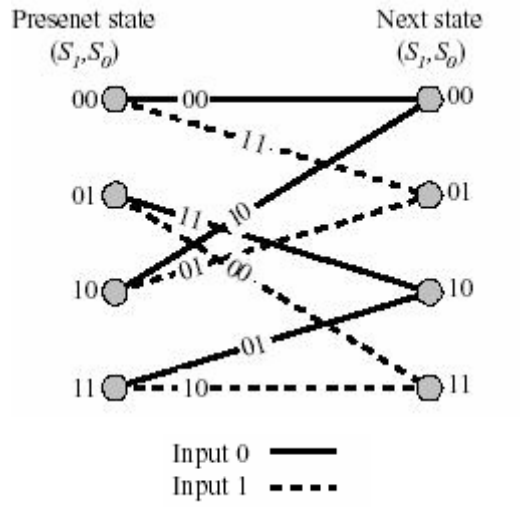


Figure 2.4: Trellis Diagram

The encoding for the sequence 0 1 1 0 1 0 0 and the output sequence is 00 11 00 01 01 11 10 is shown in the Figure 2.5.

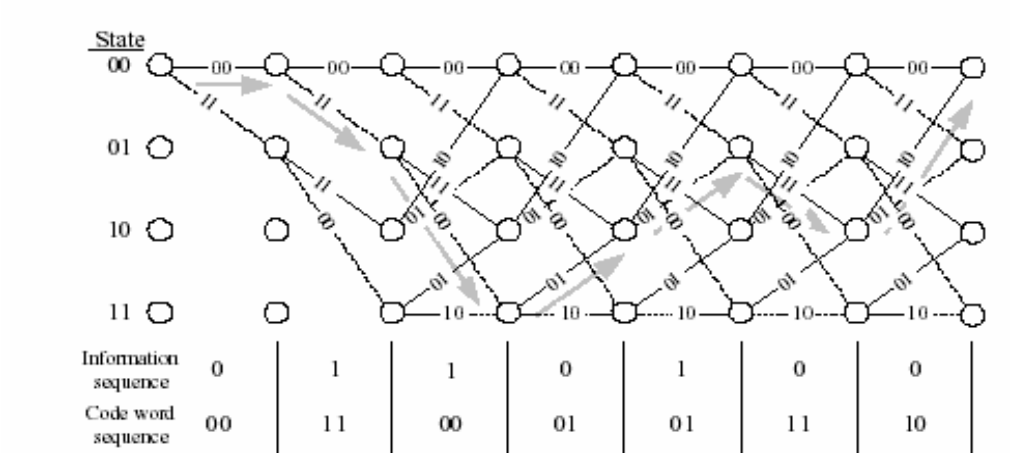


Figure 2.5: Trellis Diagram for Encoder

2.2 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 to time *n*, in terms of the shortest paths 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 [8].

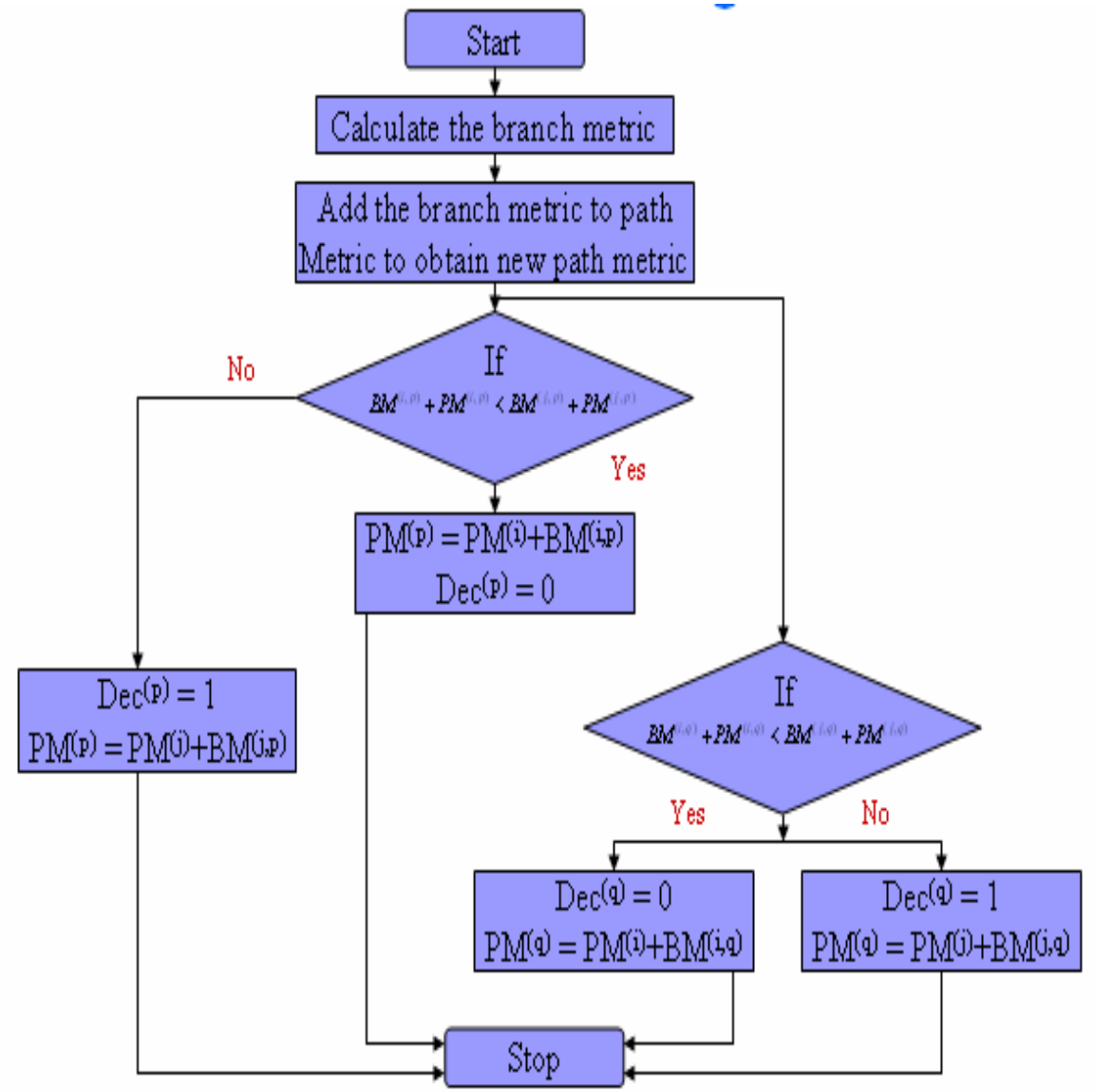


Figure 2.6: Flowchart for Viterbi Algorithm

2.3 VITERBI DECODING FOR ERROR FREE CHANNEL

The sequence of Figure 2.5 is assumed. Figure 2.7 shows the decoding sequence for two codewords received. The hamming distance between the codewords being received and the output of encoder is calculated.

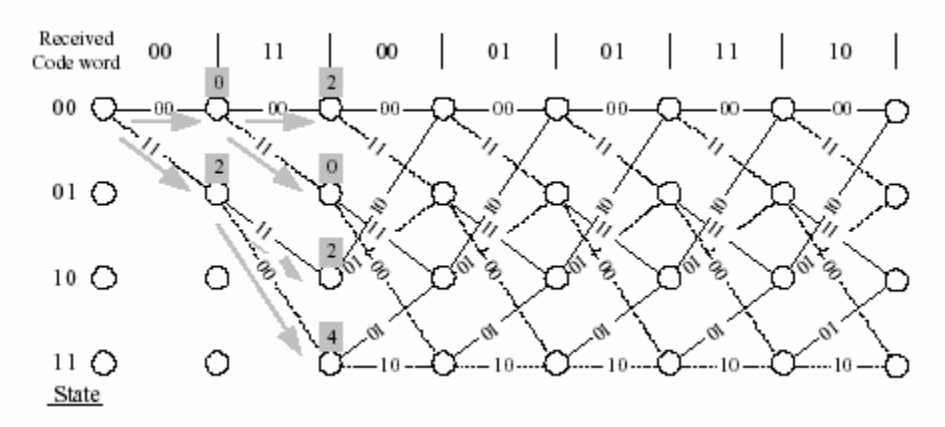


Figure 2.7: Decoding Example for Calculation of Hamming Distance

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. The same operations are done for each state.

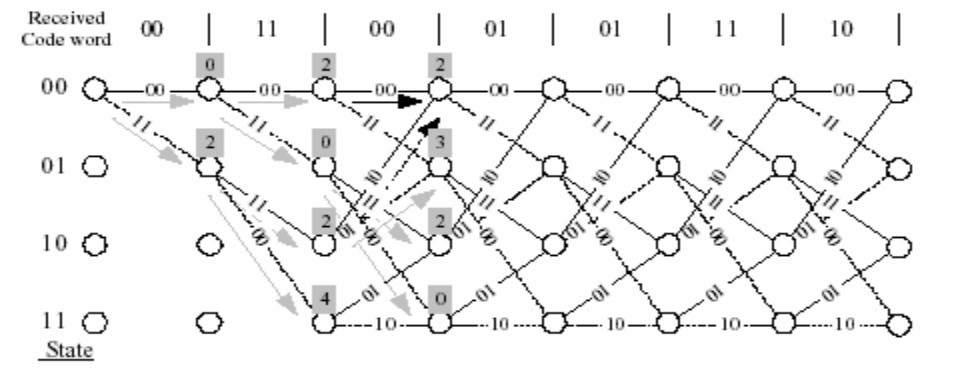


Figure 2.8: Decoding Example for Calculation of Path Metric

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 2.9.

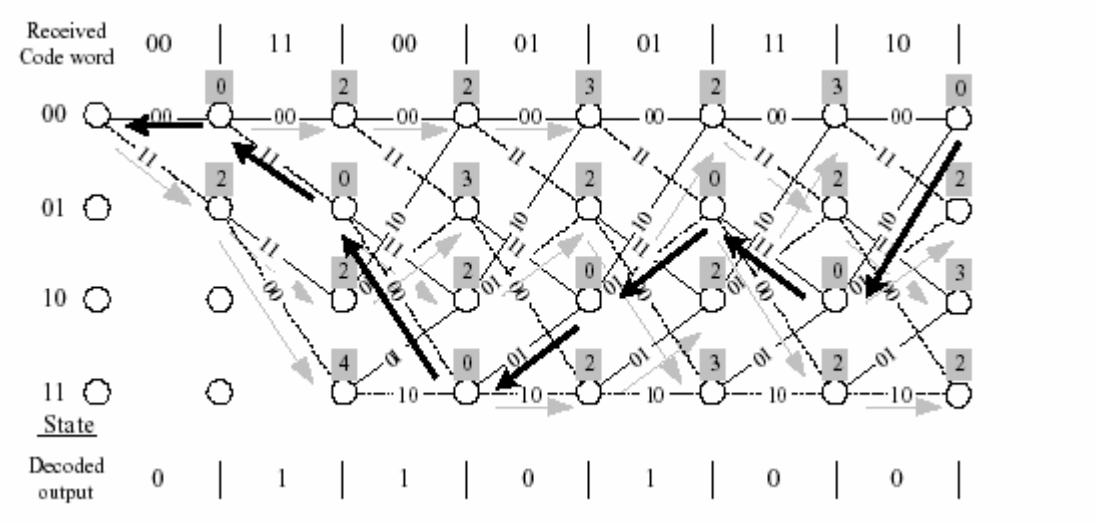


Figure 2.9: Decoding by Traceback Method

2.4 VITERBI DECODING FOR NOISY CHANNEL

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 2.10 shows first four steps of decoding process.

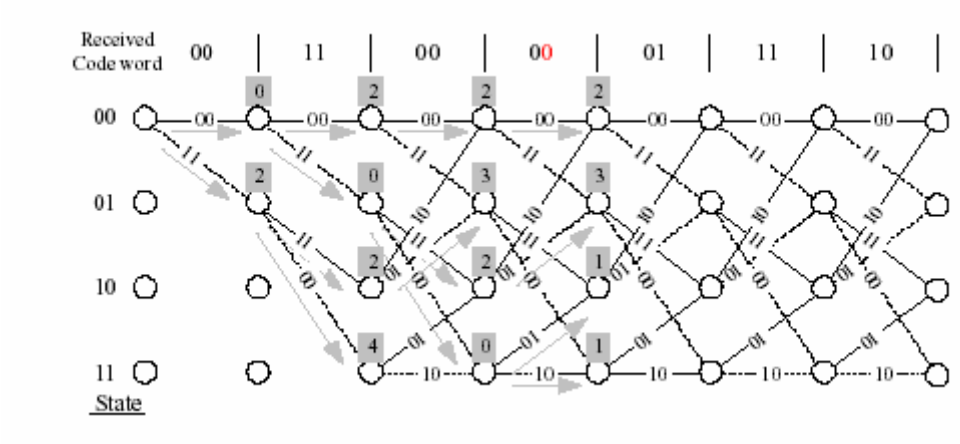


Figure 2.10: Decoding for Noisy Channel for Four Code Words

Similar steps are followed for all the codewords and complete diagram is as shown in Figure 2.11.

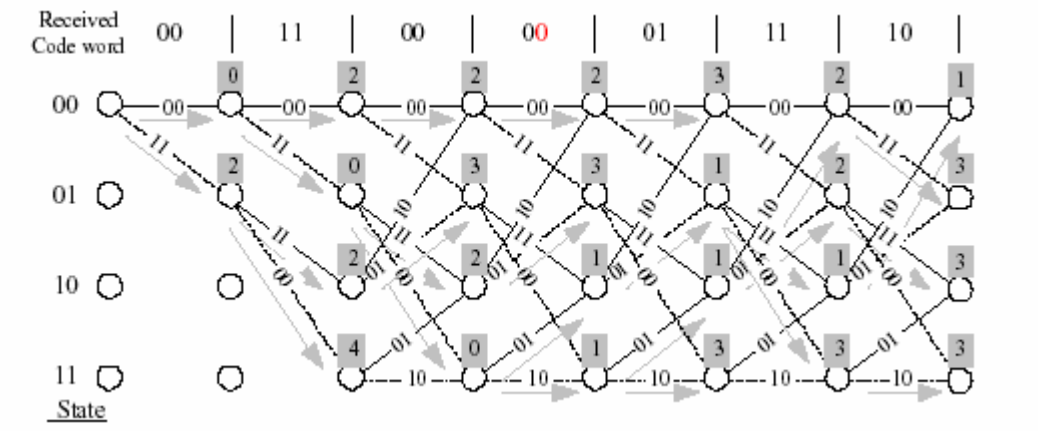
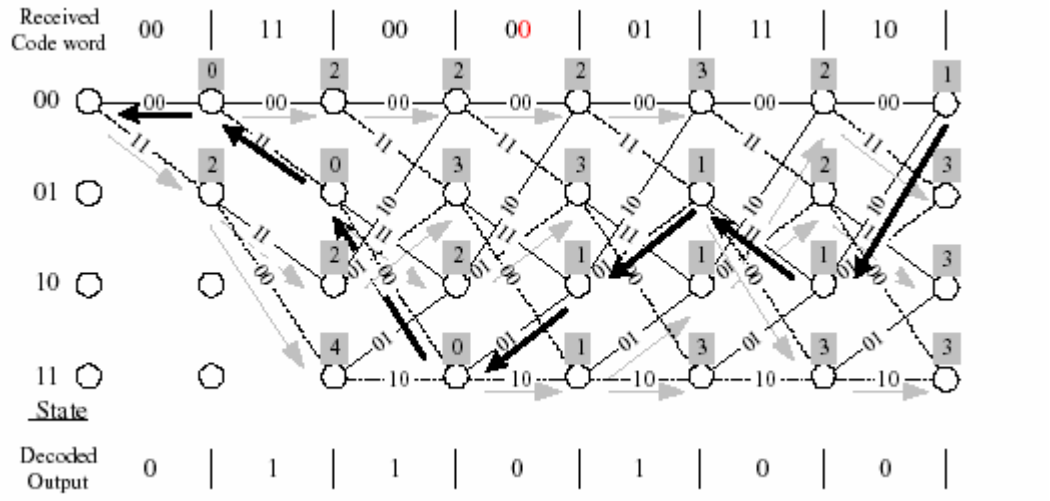


Figure 2.11: Complete Decoding Diagram



Fgure2.12: Decoding for Noisy Channel Using Traceback Method

The Figure 2.12 shows that when the final path is traced back even with one bit of error, output is correct.

2.5 VITERBI DECODER

The basic units of Viterbi decoder are branch metric unit, add compare and select unit and survivor memory management unit [9, 18].

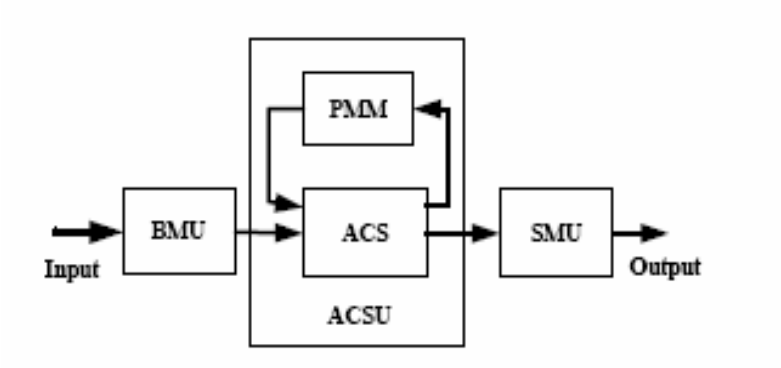


Figure 2.13: Block Diagram of Viterbi decoder

2.5.1 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.

2.5.2 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, 11, and 12].

2.5.3 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 [13].

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].

2.5.3.1 TRACEBACK METHOD

In the TB method, the storage can be implemented as RAM and is called the path memory. Comparisons in the ACS unit and not the actual survivors 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 [12].

Trace back architecture has a limited memory bandwidth in nature, and thus limits the decoding speed [5].

2.5.3.2 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 2.14. The register of state S1 at t=3 contains '101'. This is the decoded output sequence along the hold path from the initial state [13].

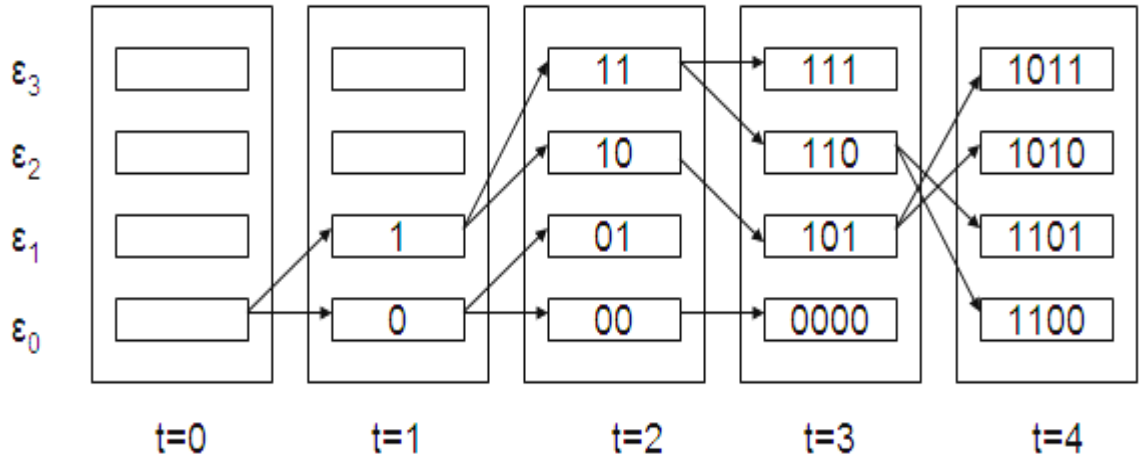


Figure 2.14: Register Exchange Method

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 [12].

2.6 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.

2.6.1 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.

2.6.2 SOFT DECISION VITERBI DECODING

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 an 8-ary output [11, 14].

2.7 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].

2.8 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. In the receiver end, a synchronized de interleaver is used which performs the inverse operation and reassembles the original code sequence before 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].

2.9 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.

Another type of sequential decoding algorithm is called a stack algorithm. In contrast to VA, this keeps track of $2^{k(K-1)}$ paths and corresponding metrics, the stack sequential decoding algorithm deals with fewer paths and their corresponding metrics. In a stack algorithm, more probable paths are ordered according to their metrics, with the path at the top of the stack having largest metric. At each step of the algorithm, only the path at the top of the stack is extended by one branch. In comparison to VA, the stack algorithm requires fewer metric computations, but this computational saving is offset to a large extent by the computations involved in reordering the stack after taking into account of every iteration.

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

2.10 APPLICATIONS OF CONVOLUTIONAL CODES

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. 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 the (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.

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.

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

Table 1: NASA Missions Convolutional Encoding Characteristics

Convolutional codes have been used in many NASA (National Aeronautics and Space Administration) missions since the l960. Table 1 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.

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. Table2 summarizes the characteristics of convolutional codes employed by DAB, DVB-S and DVB-T.

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

Table 2: Encoding Characteristics Employed in Digital Broadcasting

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 [13, 14].

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 [15].

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.

3.1 NOMENCULTAURE

The basic code selected has following specifications:

(1)	Nomenclature	:	Convolutional code with Viterbi decoding
(2)	Code rate	:	1/3 bit per symbol
(3)	Constraint length	:	7 bits
(4)	Connection vectors	:	G0 = 171 G1 = 165 G2=133

3.2 LOW POWER VITERBI DECODER

Low power Viterbi decoder with modified register exchange method is shown in Figure 3.1. ACS unit consumes most of the power. ACS unit is designed with bit serial architecture, which consumes lesser power. Butterfly concept is used further to reduce interconnections. Path metrics, which share the same input, are combined to form a butterfly. Also, modified register exchange method reduces the power to large extent [16].

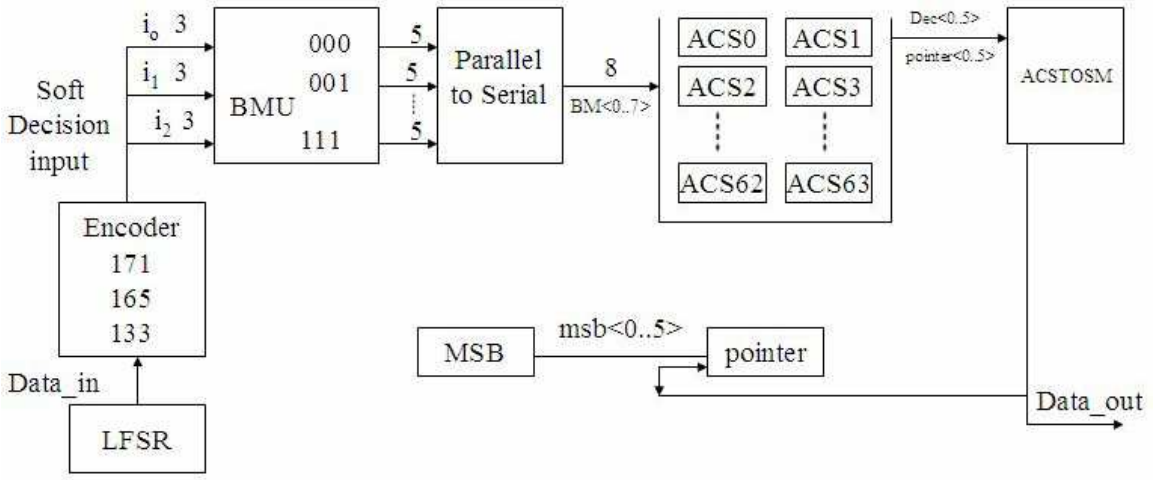


Figure 3.1: Low Power Viterbi Decoder

3.2.1 LINEAR FEEDBACK SHIFT REGISTER

LFSR produces the random input for the encoder.

3.2.2 CONVOLUTIONAL ENCODER

As constraint length is seven, six bit shit register is required. The input data is given in the serial format. Data is shifted with each clock pulse according to the generator polynomials, the inter connections between the shift register and modulo-2 adder are made. Three bit output is produced by modulo-2 addition of polynomial.

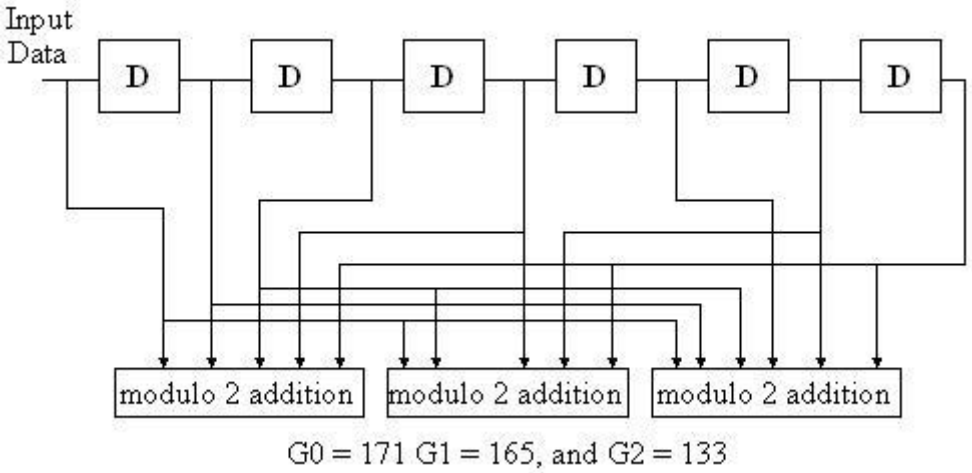


Figure 3.2: Convolutional Encoder

3.2.3 BRANCH METRIC UNIT (BMU)

This unit computes the Branch Metric (BM) for the received symbol. In case of hard decision, BM is taken as the Hamming distance between received symbol and the encoder output corresponding to a particular state transition. In case of soft decision, BM is taken as Euclidean distance.

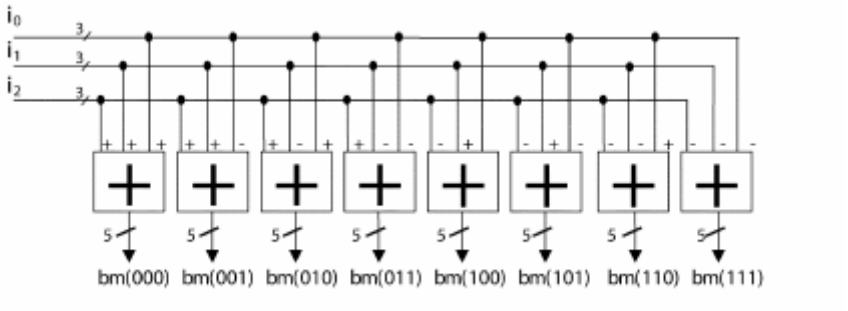


Figure 3.3: Branch Metric Unit

For three 3-bit soft decision input bits, each ranging from 0 to 3, eight 5-bit BMs are generated. The BMU performs simple add and subtract operations on the decision bits to generate the output as represented in Figure 3.3. , and the output of the BMU is still in a two's complement format. The bit serial format of the BMs is generated by the parallel to serial module at the output of the BMU, as shown in Figure 3.4, then the bit serial format BM is fed into the add-compare-select unit (ACSU) [17].

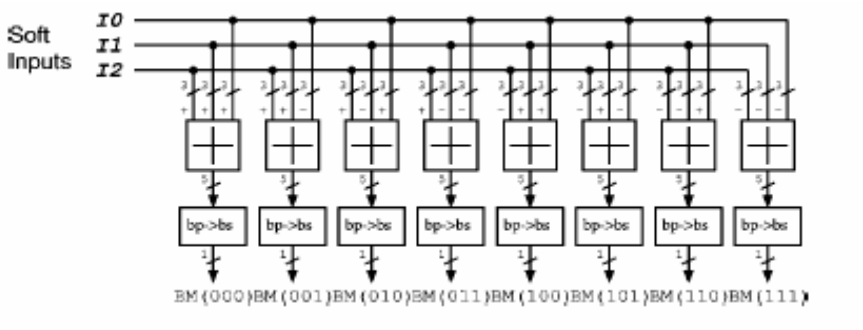


Figure 3.4: Serial Format of BM Unit

3.2.4 ADD COMPARE AND SELECT UNIT (ACSU)

The ACSU unit is composed of 64 ACS units, each is composed of an ACS butterfly module, which adds the corresponding BM to corresponding PM, compares the new PM, feeds the selected PM to ACSU unit and generates the decision bits. The decoder implements adders for computing the path metric and a compare-select section to decide on the best path. Using bit serial architecture can reduce the power of Viterbi decoder.

3.2.4.1 BIT SERIAL ACS OPERATION

Bit serial approach is used for the modified register exchange method. The bit serial ACS means that all the computations are done serially. In this decoder, 64 bit-serial (BS) ACS units including the state metric storage elements are placed in parallel. The interconnections of state metrics among the ACS units as well as the interconnection of branch metrics between the BMU and the ACS units are all implemented with a single wire, resulting in a drastic reduction of routing area compared with the bit-parallel (BP) approach. Instead of general BP arithmetic, BS arithmetic is adopted for the ACS units in our design because 64 ACS operations can be executed in parallel in moderate silicon area and power dissipation is less. The effective critical path of the BS ACS operation is shorter. In other words, it takes less time for 64 BS ACS units than BP ACS units to finish the computation of one iteration stage. Each ACS unit consists of three full-adders. Two of them are used to add the state metrics and the branch metrics, while the third one is used to compare two new state metrics. According to the decision bit, a new state metric is selected. Bit serial approach is suitable for area efficient designs and bit parallel is used for high speed system designs.

There are 128 add operations and 64 compare operations needed to be executed for each bit decoded.

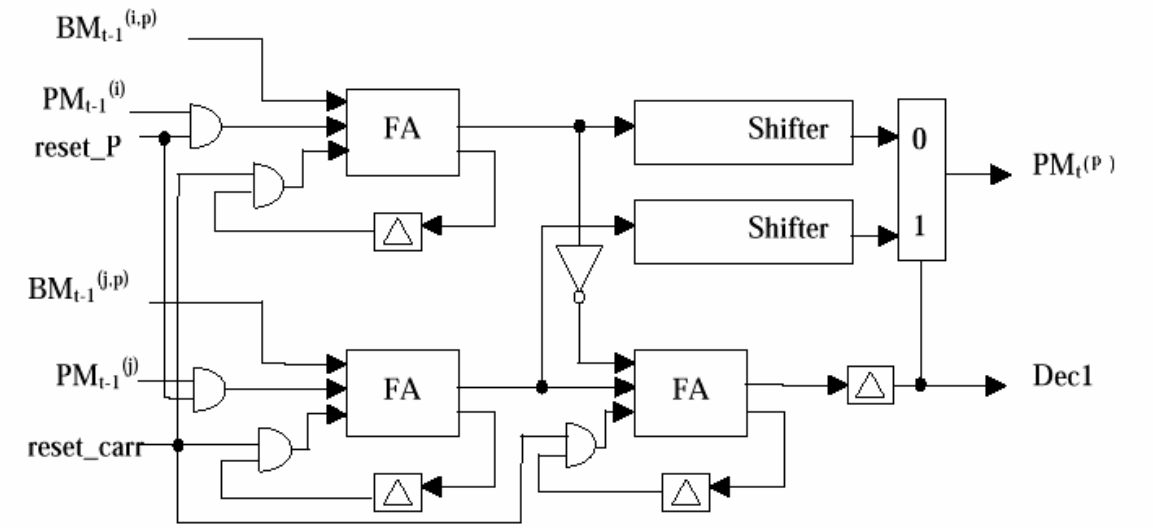


Figure 3.5: Bit Serial ACS Operations

On the other hand, the BS approach has some drawbacks. Extra storage elements are required to store state metric candidates before the decision result is obtained. Additional registers are required to store intermediate carries generated during BS addition and subtraction. There is no effective way to reduce the overhead due to the feedback carries. However, for the first drawback, area-efficient storage elements based on the ring-type first-in-first-out (FIFO) architecture can be used to reduce the overhead effectively.

The shift register based circuit consumes 35% more power than the ring-type FIFO. The total number of full adders required in the entire Viterbi decoder is 128, which represents a significant portion of the entire circuit. Therefore, the design of a low-power full adder is of great importance. The transmission gate based CMOS full adder consumes least power among different full adder architectures [18].

3.2.4.2 BUTTERFLY STRUCTURE

The two ACS processors that share the same inputs can be grouped in a Butterfly processor**.** There are 64 states so 32 butterflies are needed. Because of butterfly structure, interconnections are reduced due to which power is reduced. Figure shows the radix 2-butterfly module. The inputs j and j + N/2 are shared for the output j and 2j+1. There are two paths reaching towards each node. First is for transition 0 and other is for transition 1. When input 0 is given to current state, the next state is 2j but when input to current

state is 1, output is 2j+1.The path with lower value of accumulated path metric is assigned [25].

The branch metric and path metric of the ith path and jth path are accumulated. If the accumulated path metric of ith path is smaller, decision is made in the favor of ith path and decision bit generated is 0. The new path metric will be the accumulated path metric of ith path.

If

$$BM^{(i,p)} + PM^{(i,p)} < BM^{(j,p)} + PM^{(j,p)}$$

Then

$$Dec^{(p)} = 0$$

$$PM^{(p)} = PM^{(i,p)} + BM^{(i,p)}$$

If the accumulated path metric of jth path is smaller, decision is made in the favor of jth path and decision bit generated is 1. The new path metric will be the accumulated path metric of jth path

If

$$BM^{(i,p)} + PM^{(i,p)} > BM^{(j,p)} + PM^{(j,p)}$$

$$Dec^{(p)} = 1$$

$$PM^{(p)} = PM^{(j,p)} + BM^{(j,p)}$$

If

$$BM^{(i,q)} + PM^{(i,q)} < BM^{(j,q)} + PM^{(j,q)}$$

Then

$$Dec^{(q)} = 0$$

$$PM^{(q)} = PM^{(j,q)} + BM^{(j,q)}$$

If

$$BM^{(i,q)} + PM^{(i,q)} > BM^{(j,q)} + PM^{(j,q)}$$

Then

$$Dec^{(q)} = 1$$

$$PM^{(q)} = PM^{(j,q)} + BM^{(j,q)}$$

According to the symmetric characteristics of VD

$$BM^{(i,q)} = BM^{j,p)}$$

$$BM^{(i,p)} = BM^{j,q)}$$

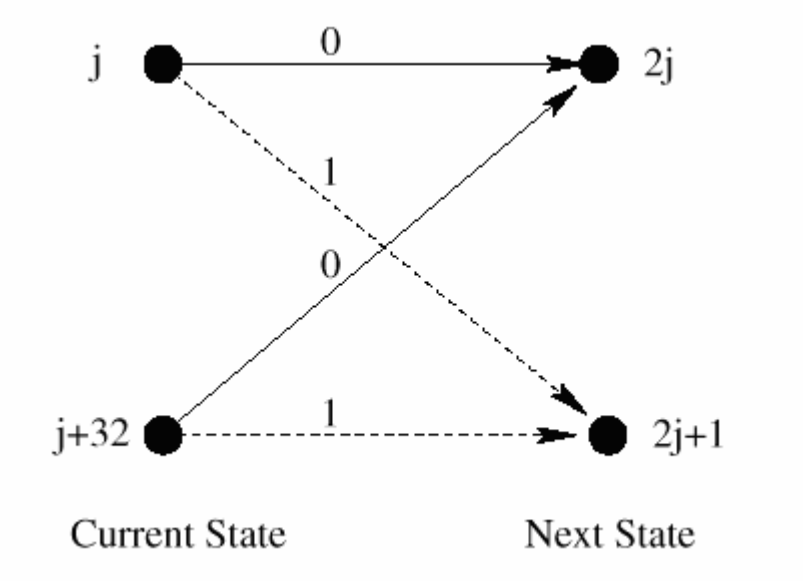


Figure 3.6: Radix 2 Butterfly Structure

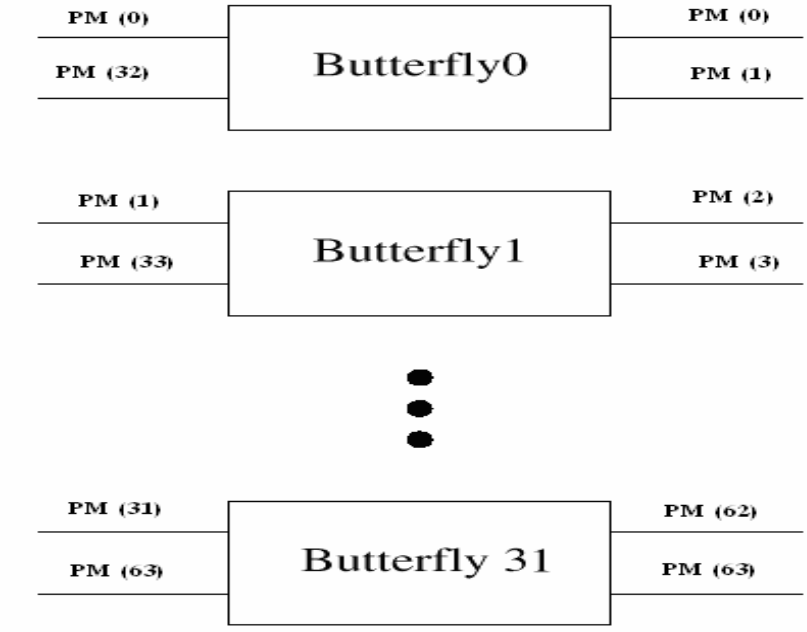


Figure 3.7: Interconnected ACS Unit

3.2.5 ACS TO SURVIVOR MEMORY MODULE

The ACS to survivor memory (ACSTOSM) is employed to route the decision of the appropriate ACS module to the output. The ACSTOSM is a 64-to-1 decoder. The select signal for this large decoder is the output of the pointer module. The output of the ACSTOSM module is the decoded output sequence of the decoder but is also fed back into the pointer module to update the current state in the decoding trellis.

3.2.6 MSB

The MSB module has eight outputs that point to the eight bits of the pointers in parallel. The reset signal causes the eighth bit of the MSB block to be high, which causes the eighth bit of any pointer to be the MSB of that pointer.

3.2.7 POINTER

The pointer block contains the current state of the decoder (6 bits). It is reset to zero, the initial state of the encoder. Then, for each bit decoded, the pointer content is updated, by the output of the ACSTOSM module. The exact position of the bit that will be updated is determined by the MSB block, which acts as a circular pointer to the pointer block [22].

3.3 REGISTER EXCHANGE METHOD

The register exchange method is used for decoding the data. The initial states are S0, S1, S2 and S3. The input data is shifted to the left and appended to the LSB each time. The transitions for different time intervals are shown in figure. At t=0, for state S0, when the input is 1, it remains in state S0 and the register contents are updated to 0. When input is 1, the state changes to S1 and the register contents of row 2 are updated to 1. At t=1, for state S0, when input is 0, the state is S0 and register contents are 00. When input is 1, the state is S1 and register contents are updated to 01. All the register contents are updated in the same way [12].

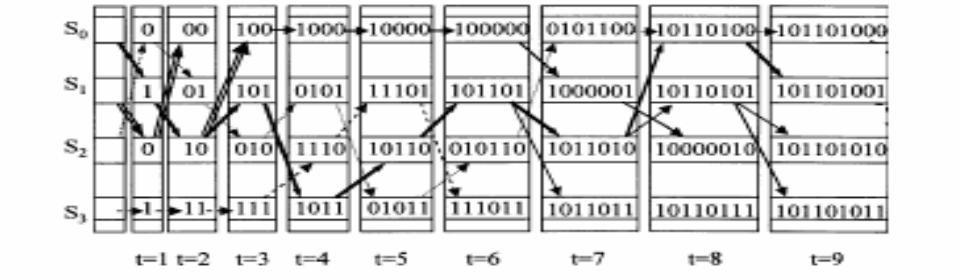


Figure 3.8: Register Exchange Method

3.4 MODIFIED REGISTER EXCHANGE METHOD

Traceback method is suited for larger constraint lengths. RE method is acceptable for trellis with only small number of states. Modified RE method is used for large constraint length VDs. In addition, the modified RE method is simpler and has lower power dissipation than that of the TB method. The RE method is based on successive RE operations between two origin states and two destination states, constructing a butterfly unit The shifting of one origin state to the left, and appending the decoded bit (the bit that causes the transition) to the least significant bit (LSB) of that origin state results in the associated destination state. This is the address of the destination register to which the origin register will be transferred. The proposed algorithm uses the "pointer" concept. Instead of moving the contents of the first register to a second register, the pointer to the first register is altered to point to the second register. In the VD, every state is assigned a register and a pointer; the pointer to the register simply carries the current state.

The pointer to register that carries the value is shifted to the left, and the bit, which causes the survivor path transition, is appended to the LSB. Then, the decoded bit is appended to the contents of the register whose pointer value is changed. It is noted that the register has a fixed physical location; only the value of its pointer changes, and a bit is appended to the corresponding register for each code word received [17].

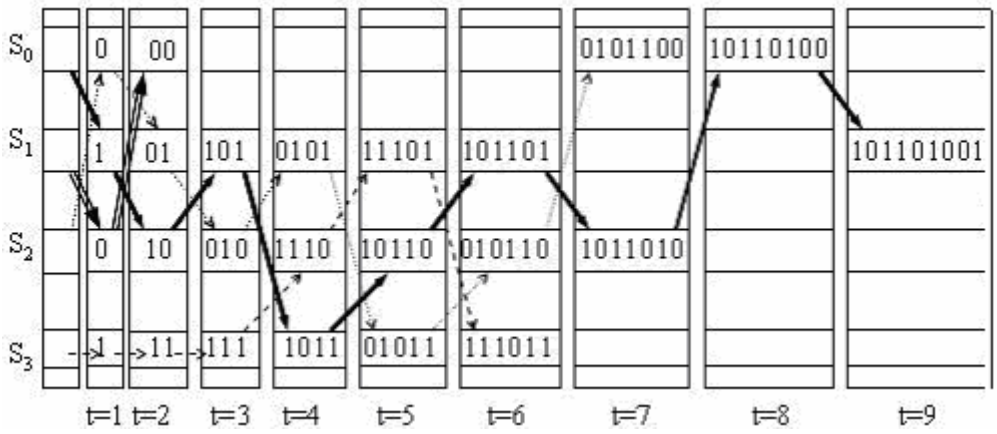


Figure 3.9: Modified RE Method

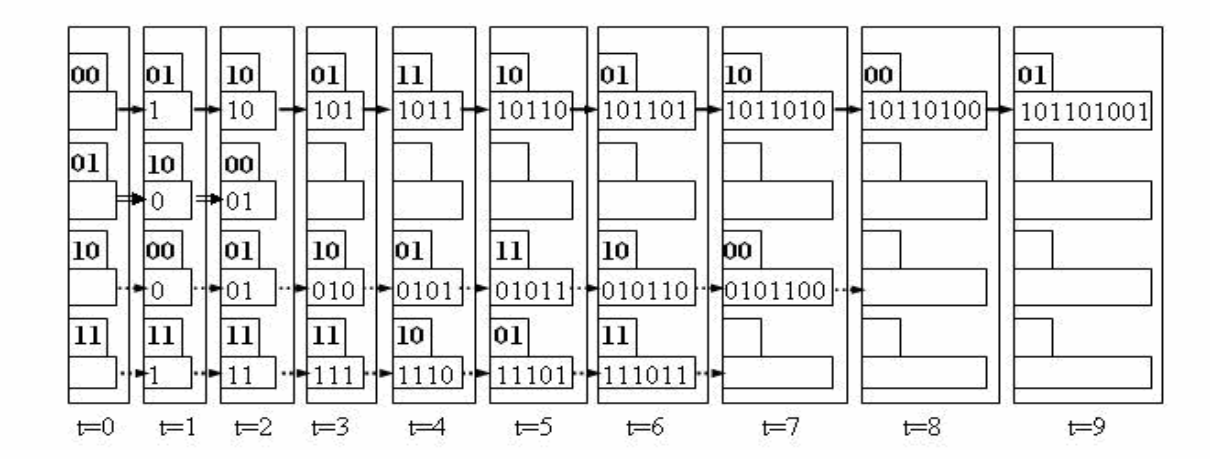


Figure 3.10: New RE Method with Pointer Implementation

The upper register carries the pointer and the lower register carries the decoded bits. If the initial state is zero, then only the first row of memory is needed. In other words, the storage of the decoded bits is necessary in order to choose only one row of memory at the end to represent the actual decoded bits. If the required row of memory is predetermined, then there is no need for the storage of the other rows. There is no need for the storage of the row that is assigned to the predetermined initial state, because the RE approach generates the decoded bits in the correct order. The decoded bits are produced, and then read out from the decoder [16].

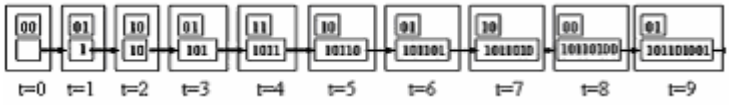


Figure 3.11: Modified RE Method with Pointer Concept

CHAPTER-4

PROGRAMMABLE LOGIC DEVICES

4.1 PROGRAMMABLE DEVICES

Programmable devices are those devices which can be programmed by the user. Various programmable devices are PLDs, CPLDs, ASICs and FPGAs.

4.1.1 PROGRAMMABLE LOGIC DEVICES

At the low end of the spectrum are the original Programmable Logic Devices (PLDs). A programmable logic device is an IC that is user configurable and is capable of implementing logic functions. These were the first chips that could be used to implement a flexible digital logic design in hardware. In other words, you could remove a couple of the 7400-series TTL parts (ANDs, ORs, and NOTs) from your board and replace them with a single PLD. Other names for this class of device are Programmable Logic Array (PLA), Programmable Array Logic (PAL), and Generic Array Logic (GAL).

PLDs have several clear advantages over the 7400-series TTL parts that they replaced. First, of course, is that chip requires less board area, power, and wiring. Another advantage is that the design inside the chip is flexible, so a change in the logic doesn't require any rewiring of the board. Rather, simply replacing that one PLD with another part that has been programmed with the new design can alter the decoding logic.

Inside each PLD is a set of fully connected macrocells. These macrocells are typically comprised of some amount of combinatorial logic (AND and OR gates) and a flip-flop. In other words, a small Boolean logic equation can be built within each macrocell. Hardware designs for these simple PLDs are generally written in languages like ABEL or PALASM (the hardware equivalents of assembly) or drawn with the help of a schematic capture tool.

4.1.2 COMPLEX PROGRAMMABLE DEVICES

As chip density is increased, it was natural for the PLD manufacturers to evolve their products into larger (logically, but not necessarily physically) parts called Complex Programmable Logic Devices (CPLDs). For most practical purposes, CPLDs can be thought of as multiple PLDs (plus some programmable interconnect) in a single chip. The larger size of a CPLD allows you to implement either more logic equations or a more

complicated design. In fact, these chips are large enough to replace dozens of those 7400-series parts.

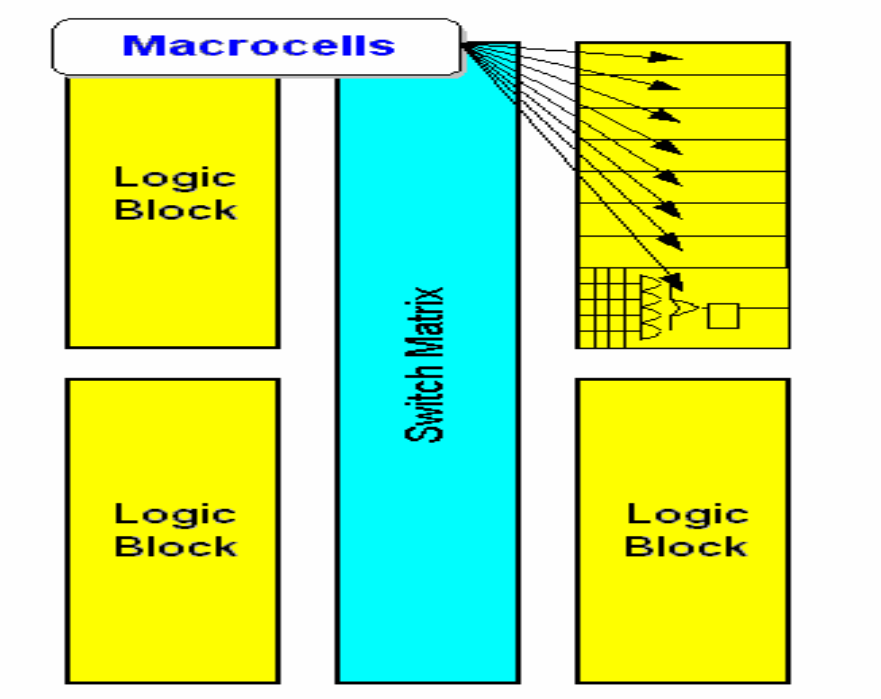


Figure 4.1: Internal Structure of a CPLD

Figure 4.1 contains a block diagram of a CPLD. Each of the four logic blocks shown there is the equivalent of one PLD. However, in an actual CPLD there may be more or less than four logic blocks. These logic blocks are themselves comprised of macrocells and interconnect wiring, just like an ordinary PLD. Unlike the programmable interconnect within a PLD, the switch matrix within a CPLD may or may not be fully connected. In other words, some of the theoretically possible connections between logic block outputs and inputs may not actually be supported within a given CPLD. The effect of this is most often to make 100% utilization of the macrocells very difficult to achieve. Some hardware designs simply won't fit within a given CPLD, even though there are sufficient logic gates and flip-flops available.

Because CPLDs can hold larger designs than PLDs, their potential uses are more varied. They are still sometimes used for simple applications like address decoding, but more often contain high-performance control-logic or complex finite state machines. At the

high-end (in terms of numbers of gates), there is also a lot of overlap in potential applications with FPGAs. Traditionally, CPLDs have been chosen over FPGAs whenever high-performance logic is required. Because of its less flexible internal architecture, the delay through a CPLD (measured in nanoseconds) is more predictable and usually shorter.

4.1.3 FIELD PROGRAMMABLE GATE ARRAYS

'Field Programmable' means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device. A typical integrated circuit performs a particular function defined at the time of manufacture. In contrast, a program written by someone other than the device manufacturer defines the FPGA's function. Depending on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up. This user programmability gives the user access to complex integrated designs without the high engineering costs associated with application specific integrated circuits (ASIC). The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. The individual cells are interconnected by a matrix of wires and programmable switches.

The logic cell architecture varies between different device families. Generally speaking, each logic cell combines a few binary inputs (typically between 3 and 10) to one or two outputs according to a boolean logic function specified in the user program. The cell's combinatorial logic may be physically implemented as a small look-up table memory (LUT) or as a set of multiplexers and gates. LUT devices tend to be a bit more flexible and provide more inputs per cell than multiplexer cells at the expense of propagation delay.

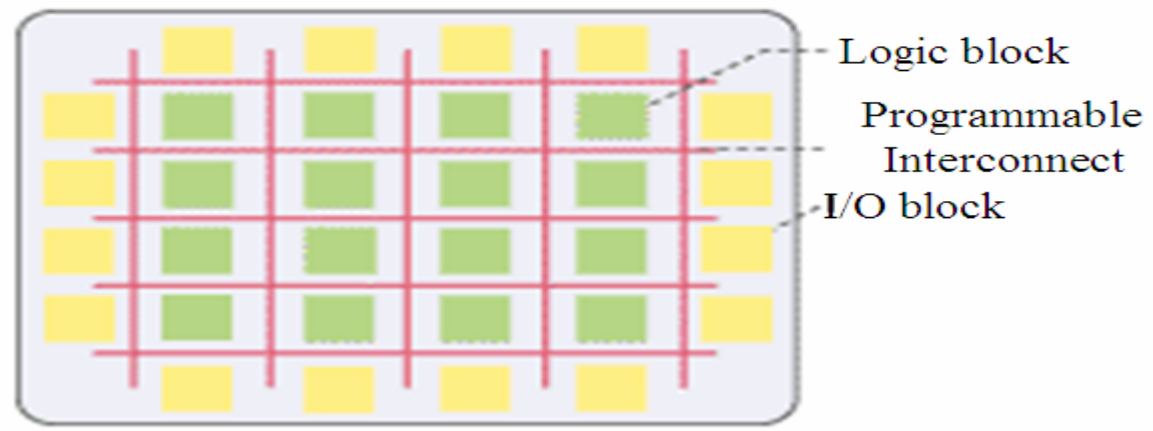


Figure 4.2: Internal Structure of an FPGA

The development of the FPGA was distinct from the PLD/CPLD evolution. There are three key parts of its structure: logic blocks, interconnect, and I/O blocks. The I/O blocks form a ring around the outer edge of the part. Each of these provides individually selectable input, output, or bi-directional access to one of the general-purpose I/O pins on the exterior of the FPGA package. Inside the ring of I/O blocks lies a rectangular array of logic blocks. The wire connecting logic block to logic blocks and I/O to logic block is called as programmable inter connect.

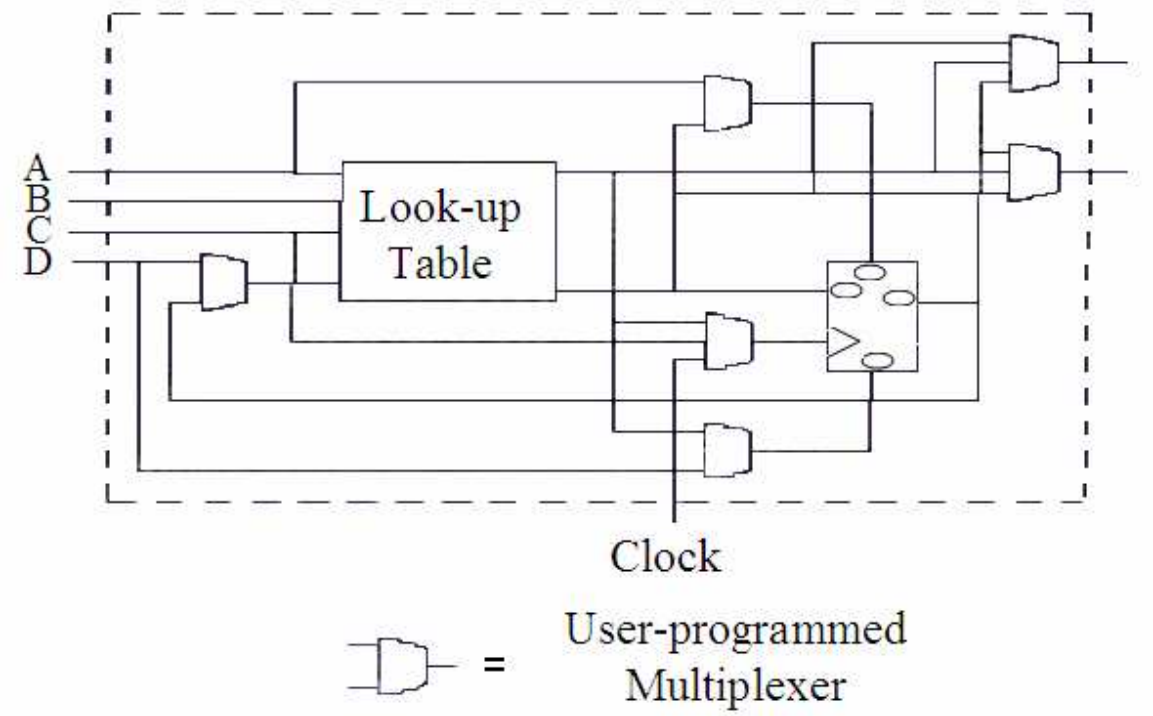


Figure 4.3: Internal Architecture of CLB

The logic blocks within an FPGA can be as small and simple as the macrocells in a PLD (a so-called fine-grained architecture) or larger and more complex (coarse-grained). However, they are never as large as an entire PLD, as the logic blocks of a CPLD are. The logic blocks of a CPLD contain multiple macrocells. But the logic blocks in an FPGA are generally nothing more than a couple of logic gates or a look-up table and a flip-flop.

4.1.3.1 ADVANTAGES OF FPGA

Because of all the extra flip-flops, the density is higher from several thousand gates to few million gates and the architecture of an FPGA is much more flexible than that of a CPLD. This makes FPGAs better in register-heavy applications. They are also often used in place where the processing of input data streams must be performed at a very fast pace. In addition, FPGAs are usually denser (more gates in a given area) and cost less than CPLD, so they are the best choice for larger logic designs. FPGA's uses static memory so they are reprogrammable [15].

4.2 HARDWARE DESIGN AND DEVELOPMENT

A description of the hardware's structure and behavior is written in a high-level hardware description language (usually VHDL or Verilog) and that code is then compiled and downloaded prior to execution. Of course, schematic capture is also an option for design entry, but it has become less popular as designs have become more complex and the language-based tools have improved. The overall process of hardware development for programmable logic is shown in Figure 4.4 [19].

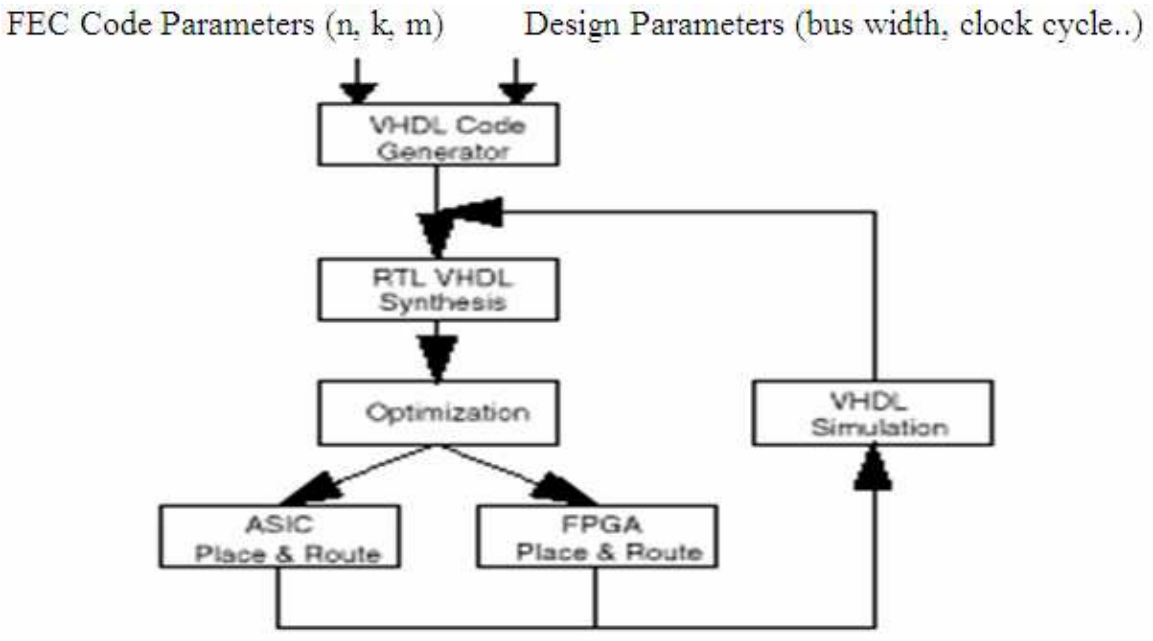


Figure 4.4: Design Flow

4.2.1 DESIGN ENTRY

In the design entry process, the behavior of circuit is written in hardware description language like VHDL or Verilog.

4.2.2 SYNTHESIS

First, an intermediate representation of the hardware design is produced. This step is called synthesis and the result is a representation called a netlist. In this step, any semantic and syntax errors are checked. The synthesis report is created which gives the details of errors and warning if any. The netlist is device independent, so its contents do not depend on the particulars of the FPGA or CPLD; it is usually stored in a standard format called the Electronic Design Interchange Format (EDIF).

4.2.3 SIMULATION

Simulator is a software program to verify functionality of a circuit. The functionality of code is checked. The inputs are applied and corresponding outputs are checked. If the expected outputs are obtained then the circuit design is correct. Simulation gives the output waveforms in form of zeros and ones. Although problems with the size or timing of the hardware may still crop up later, the designer can at least be sure that his logic is functionally correct before going on to the next stage of development.

4.2.4 IMPLEMENTATION

Device implementation is done to put a verified code on FPGA. The various steps in design implementation are:

1. Translate
2. Map
3. Place and route
4. Configure

4.2.4.1 TRANSLATE

Translate converts the EDIF file to the NGD (Native Generic Description File) which means code is converted to the gates or net lists. The translate process generates the translate report which gives the errors and warnings in translation process. This report also gives the list of device and I/O utilization, which helps the designer to determine the selection of best device.

4.2.4.2 MAP

Mapping converts the NGD (Native Generic Description) file obtained from translate process to the NCD (Native Circuit Description File) which means the gates are converted to the physical components like flip flops and multiplexer.

4.2.4.3 PLACE AND ROUTE

Place is the process of selecting specific logic blocks in the FPGAs where design gates will reside. Route is the physical routing of interconnect between logic blocks. This means that logic blocks, CLB, I/O blocks are assigned to specific locations on die and interconnections are made between them. This step involves mapping the logical structures described in the net list onto actual macrocells, interconnections, and input and output pins. This process is similar to the equivalent step in the development of a printed circuit board, and it may likewise allow for either automatic or manual layout optimizations. The result of the place & route process is a bitstream. This name is used generically, despite the fact that each CPLD or FPGA (or family) has its own, usually proprietary, bitstream format. Bitstream is the binary data that must be loaded into the FPGA or CPLD to cause that chip to execute a particular hardware design.

4.3 DEVICE PROGRAMMING

Once bitstream file is created for a particular FPGA or CPLD, it is downloaded on the device. The details of this process are dependent upon the chip's underlying process technology. Programming technologies used are PROM (for one-time programmable), EPROM, EEPROM, and Flash.

Just like their memory counterparts, PROM and EPROM-based logic devices can only be programmed with the help of a separate piece of lab equipment called a device programmer. On the other hand, many of the devices based on EEPROM or Flash technology are in-circuit programmable. In other words, the additional circuitry that's required to perform device reprogramming is provided within the FPGA or CPLD silicon as well. This makes it possible to erase and reprogram the device internals via a JTAG interface or from an on-board embedded processor. In addition to non-volatile technologies, there are also programmable logic devices based on SRAM technology. In such cases, the contents of the device are volatile. This has both advantages and disadvantages. The obvious disadvantage is that the internal logic must be reloaded after every system or chip reset. That means we need an additional memory chip of some sort in which to hold the bitstream. But it also means that the contents of the logic device can be changed [20].

4.4 VHDL

VHDL is the VHSIC Hardware Description Language. VHSIC is an abbreviation for Very High Speed Integrated Circuit. It can describe the behaviour and structure of electronic systems, but is particularly suited as a language to describe the structure and behaviour of digital electronic hardware designs, such as ASICs and FPGAs as well as conventional digital circuits. Simulation and synthesis are the two main kinds of tools which operate on the VHDL language. VHDL does not constrain the user to one style of description. VHDL allows designs to be described using any methodology - top down or bottom up. VHDL can be used to describe hardware at the gate level or in a more abstract way.

4.4.1 BRIEF HISTORY OF VHDL

The development of VHDL was initiated in 1981 by the United States Department of Defence to address the hardware life cycle crisis. The cost of reprocuring electronic

hardware as technologies became obsolete was reaching crisis point, because the function of the parts was not adequately documented, and the various components making up a system were individually verified using a wide range of different and incompatible simulation languages and tools. The requirement was for a language with a wide range of descriptive capability that would work the same on any simulator and was independent of technology or design methodology.

The initial version of VHDL, designed to IEEE standard 1076-1987, included a wide range of data types, including numerical (integer, real), logical (bit and boolean), character and time, arrays of bit called bit vector and of `character` called string.

A problem not solved by this edition, however, was "multi-valued logic", where a signal's drive strength (none, weak or strong) and unknown values are also considered. This required IEEE standard 1164 which defined the 9-value `standard logic`.

The second issue of IEEE 1076 in 1993, made the syntax more consistent, allowed more flexibility in naming, added the `Xnor` operator, etc.

More recently, the language has been extended by introducing `signed` and `unsigned` types to facilitate arithmetical operations, analog and mixed-signal circuit design extensions, VITAL (VHDL Initiative Towards ASIC Libraries), and microwave circuit design extensions .

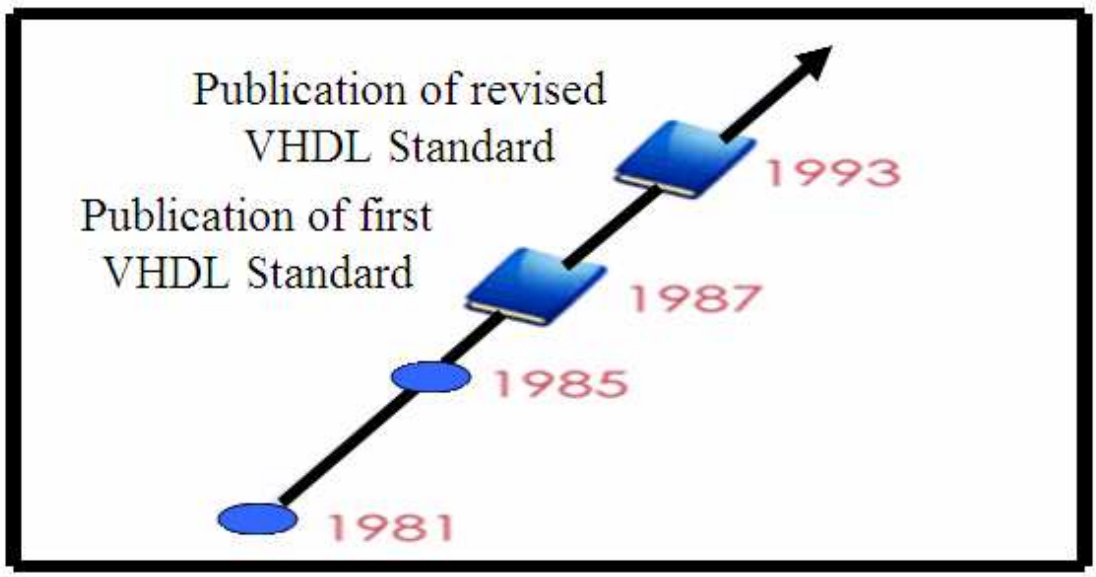


Figure 4.5: History of VHDL

4.5 FPGA IN DSP APPLICATIONS

Application Specific Integrated Circuits (ASICs) provide a flexible platform for designers to apply digital signal processing (DSP) algorithms to several application areas such as telecommunication, instrumentation, image processing and digital audio. Current FPGA technology provides about 15-30 kgates of logic capacity in a single chip with upto 10-15 MHZ sampling rates in DSP applications depending on system architecture and word length. Thus they are basically targeted to small or medium size systems or components working at maximum on the operating speed required for digital video and multimedia applications. Today, FPGA technologies are heading towards higher arithmetic capacity and performance. Also, many new technologies provide internal high-speed memories with a reasonable amount of storage capacity. Also, current trend towards larger logic modules and better long and medium range routing structure improve FPGAs capability to implement complex algorithms. Due to these trends of FPGA families' suitability for DSP applications have been improving. For DSP algorithm implementation multipliers and adders are most interesting. Currently, FPGAs provide 8×8 multipliers about 25-30 MHZ and 16-accumulators about 80-100 MHZ operating speeds. Also, it is possible to implement at maximum upto 24k internal RAMs in a single FPGA device. FPGAs reusability and high parallelism have made them more attractive for DSP applications [20,21].

5. RESULTS

All the modules are designed in VHDL and simulated in Model Sim.

5.1 SIMULATION AND SYNTHESIS RESULTS

The different modules in the Viterbi decoder design are:

(1) LFSR

(2) Encoder

(3) BMU

(4) Parallel to serial

(5) ACS

(6) ACSTOSM

(7) MSB

(8) Pointer

(9) Viterbi

These modules are designed using VHDL and synthesized using Xilinx Integrated software environment (ISE). The design is simulated using Model Sim. The design implementation is done on Xilinx Spartan 2E xc2s15-6cs144.

5.1.1 SIMULATION RESULTS OF LFSR

LFSR is used to generate the random data. The reset signal and clock signal are the inputs to LFSR and the output is the random data generation data_out.

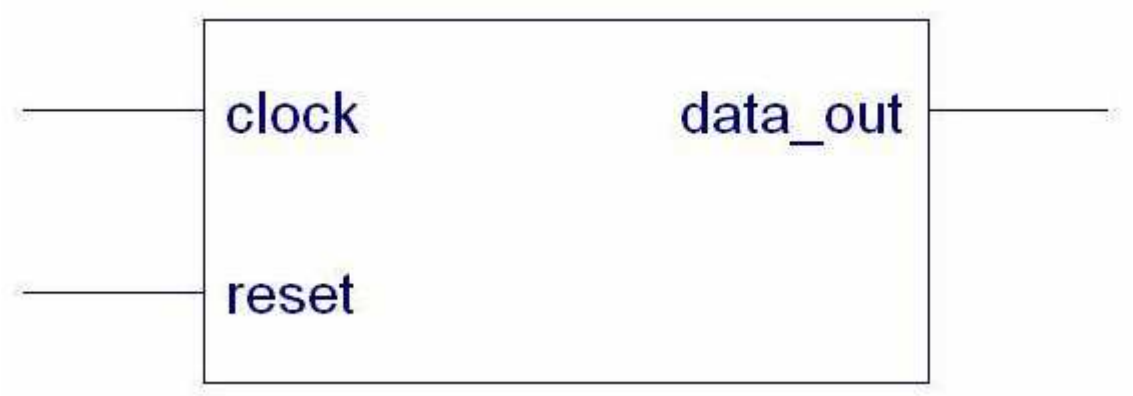


Figure 5.1: RTL of LFSR

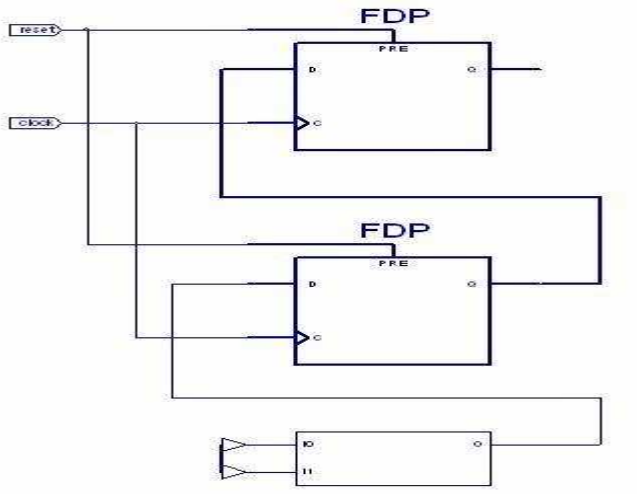


Figure 5.2: Schematic of LFSR

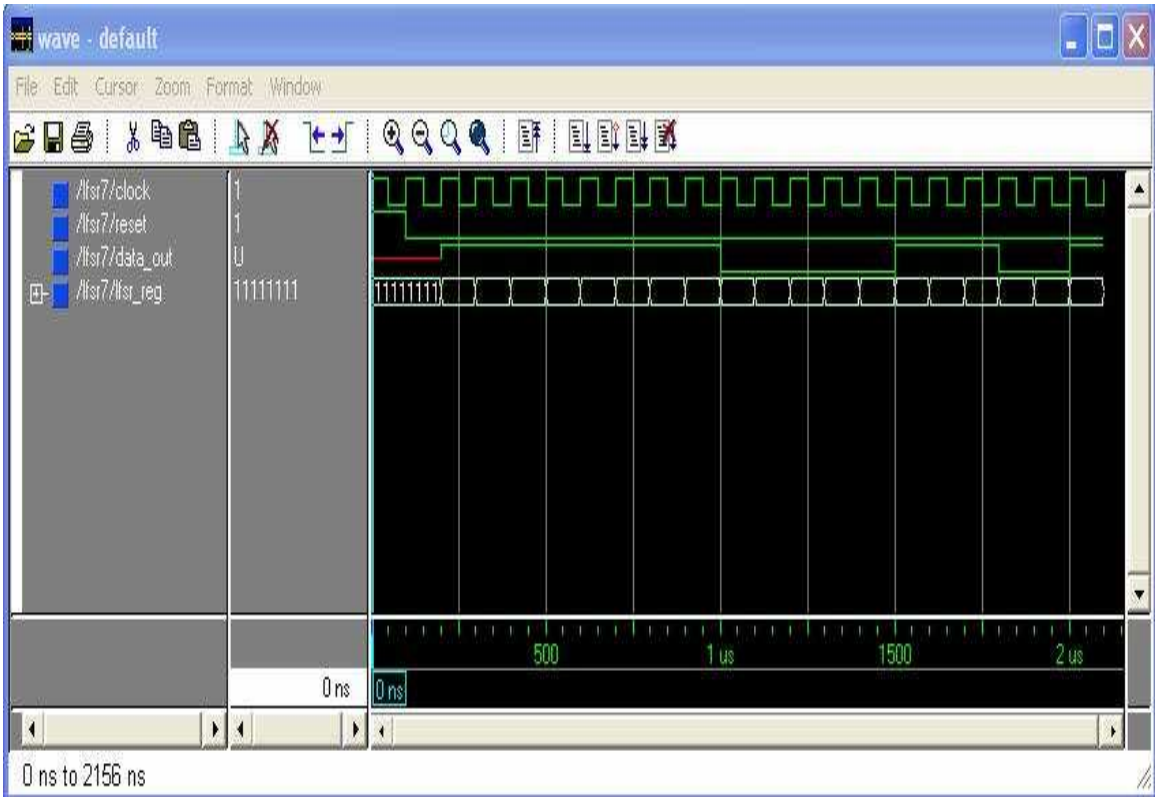


Figure 5.3: Simulations of LFSR

5.1.2. SIMULATION A ND SYNTHESIS RESULTS OF ENCODER

The second module is encoder. The outputs from the LFSR are fed to the encoder. The inputs to the module are reset, clk and data_in. The outputs generated are i0, i1, and i2 each of three bits.

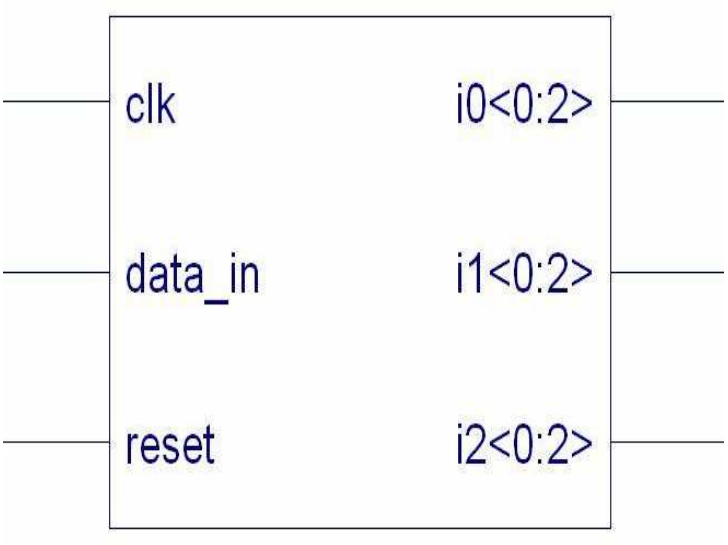


Figure 5.4: RTL of Encoder

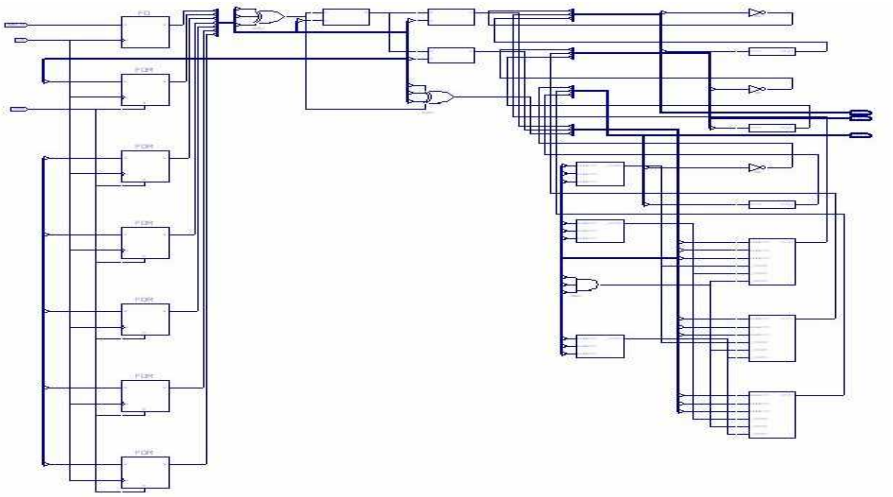


Figure 5.5: Schematic of Encoder

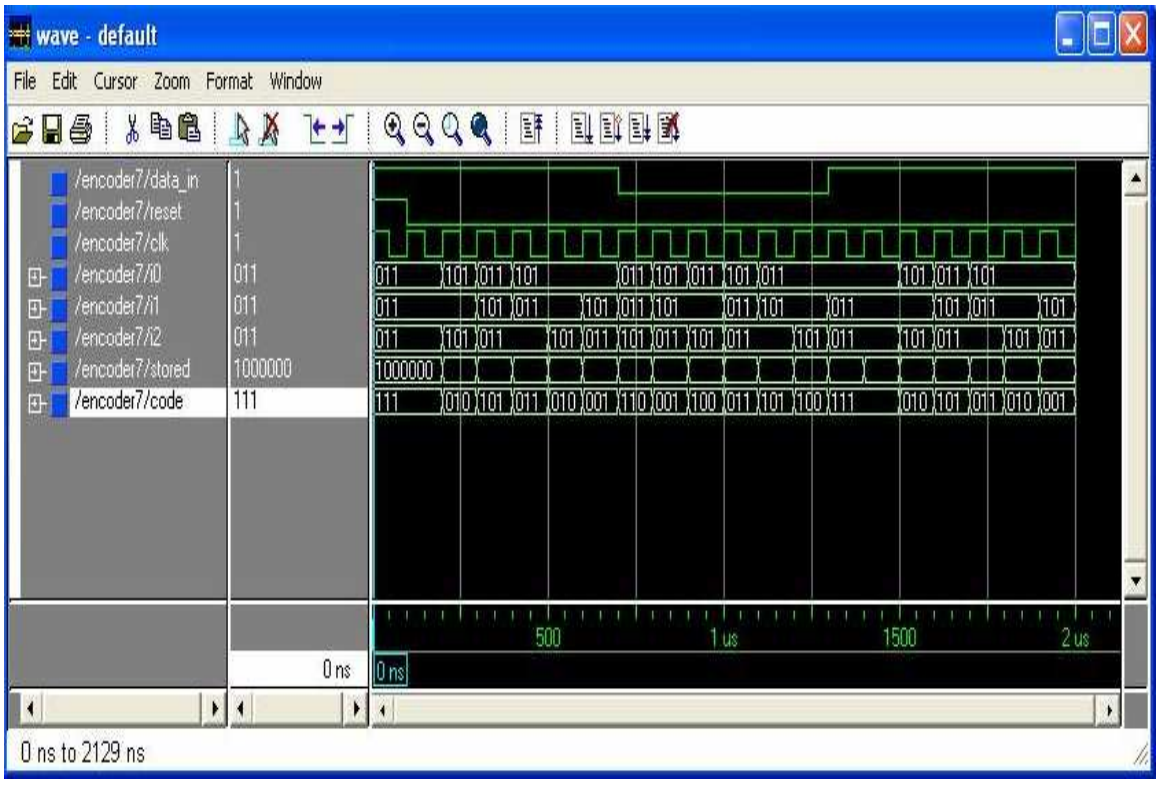


Figure5.6: Simulations of Encoder

5.1.3. SIMULATION A ND SYNTHESIS RESULTS OF BMU

The outputs from the encoder are fed to the BMU. The input signals are i0, i2 and i3 and the output signals of BMU are bmu000, bmu001, bmu010, bmu011, bmu100, bmu101, bmu110 and bmu111. Eight branch metrics each of five bits are generated from the 3-bit input of encoder.

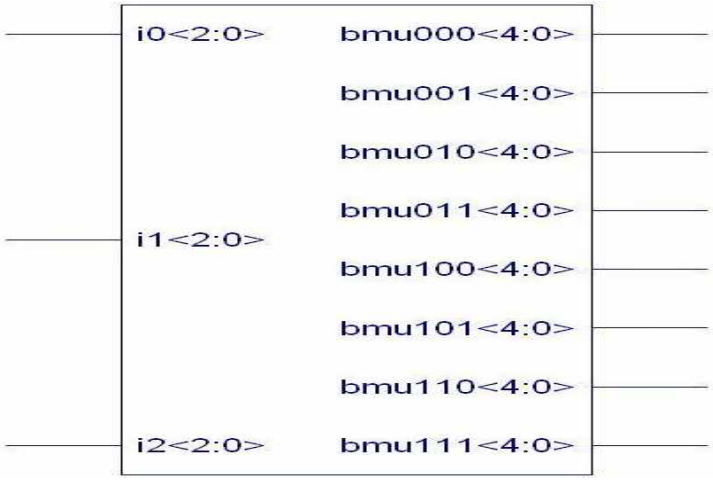


Figure 5.7: RTL of BMU

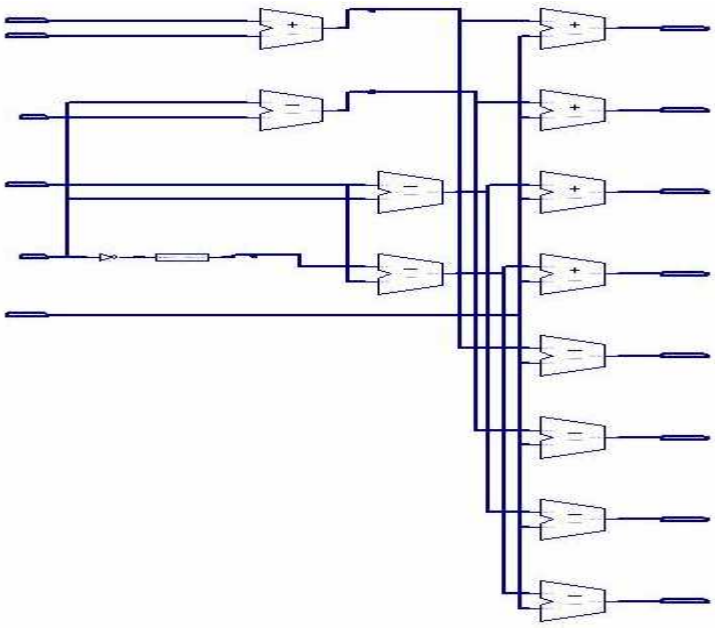


Figure 5.8: Schematic of BMU

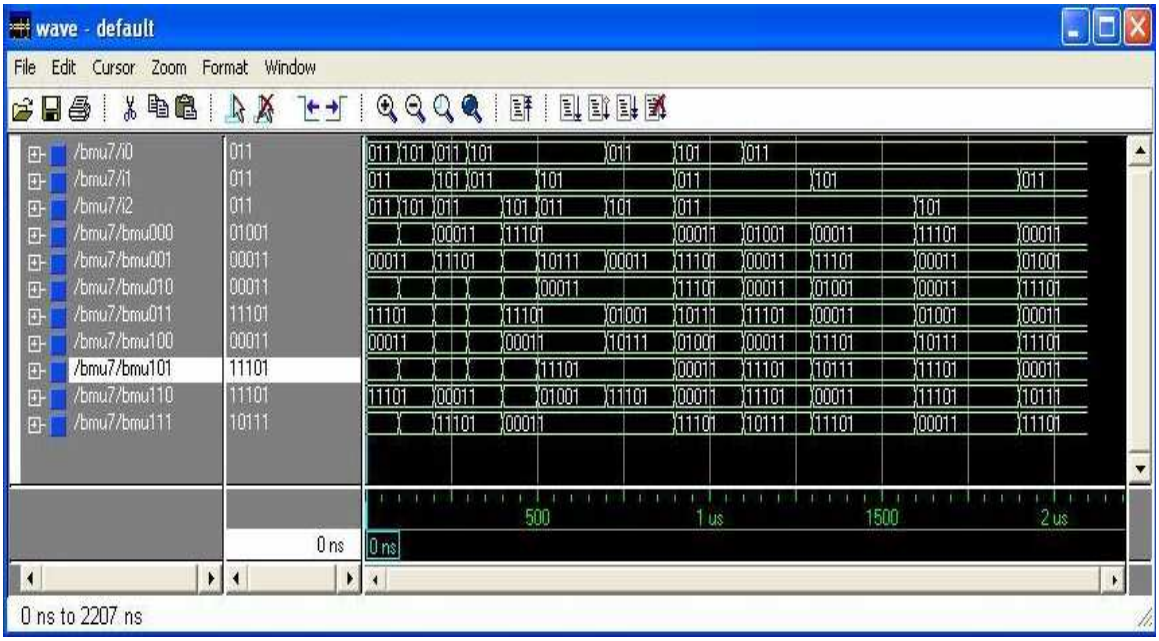


Figure 5.9: Simulations of BMU

5.1.4. SIMULATION A ND SYNTHESIS RESULTS OF PARALLEL TO SERIAL MODULE

The outputs of BM unit are fed as input to this module. This module converts the parallel format of BM to serial format. The inputs are bmu000, bmu001, bmu010, bmu011, bmu100, bmu101, bmu110 and bmu111. Each is of 5 bits and the output is BM, which is of 8 bits. This BM serves as input to next module i.e. ACS module.

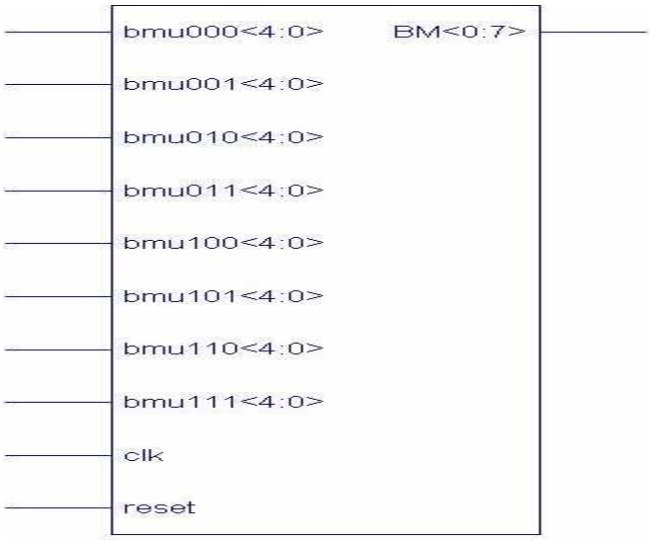


Figure 5.10: RTL of Parallel to Serial

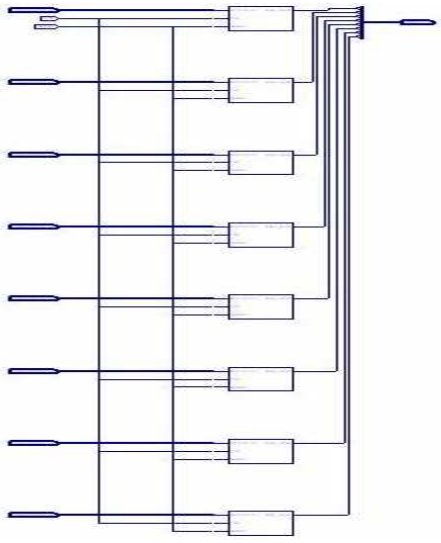


Figure 5.11: Schematic of Parallel to Serial

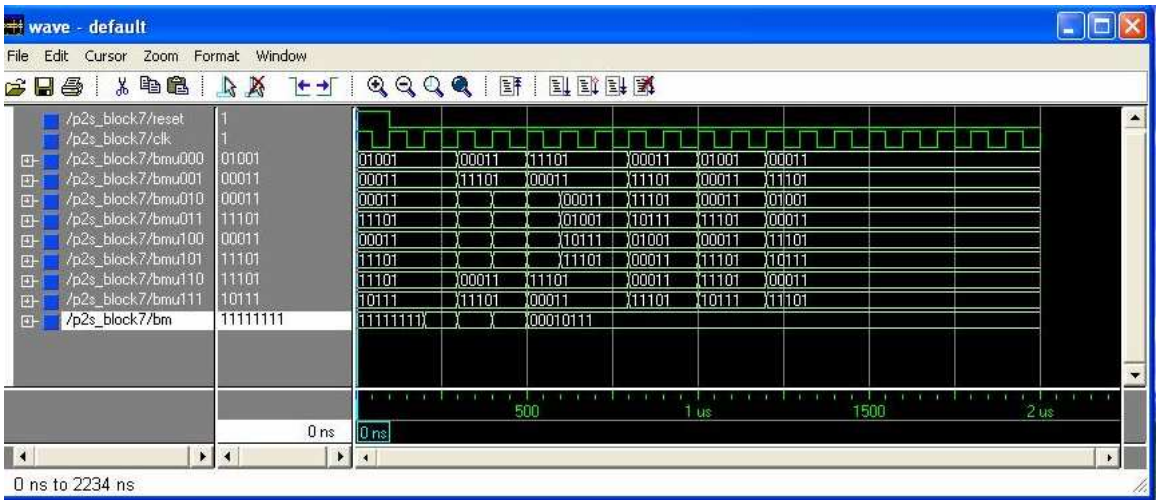


Figure 5.12: Simulation of Parallel to Serial

5.1.5. SIMULATION A ND SYNTHESIS RESULTS OF ACS MODULE

The input signals are bm, msb_bmjp, clk, reset_carry, comp_enable and reset_pm. The inputs of parallel to serial module are fed to this module The output signals are term, dec and PM. The outputs of this module are served as input to the ACSTOM module.

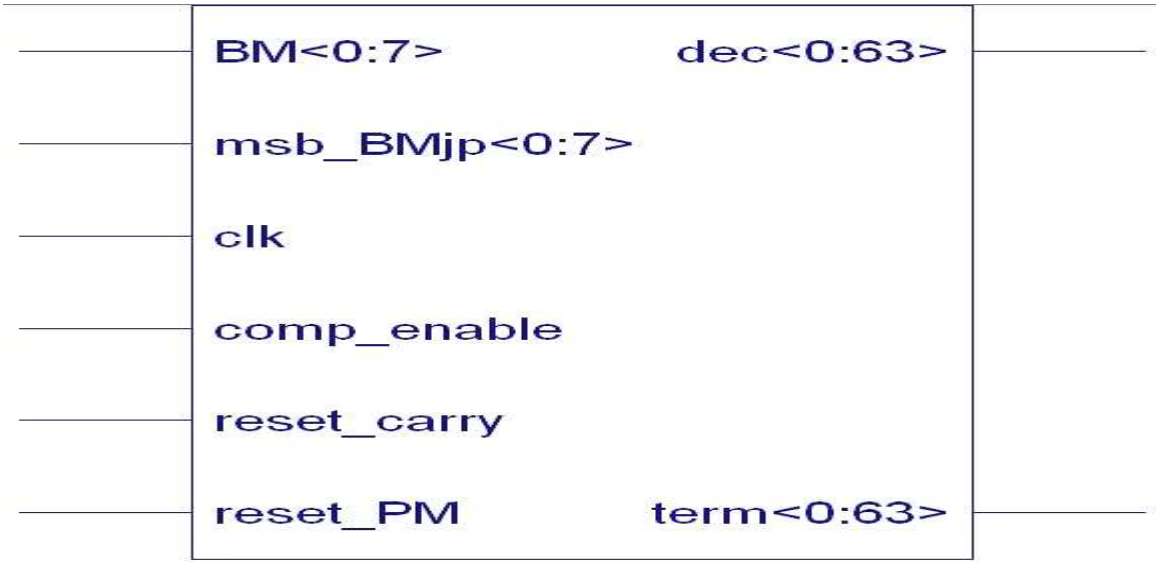


Figure 5.13: RTL of ACS

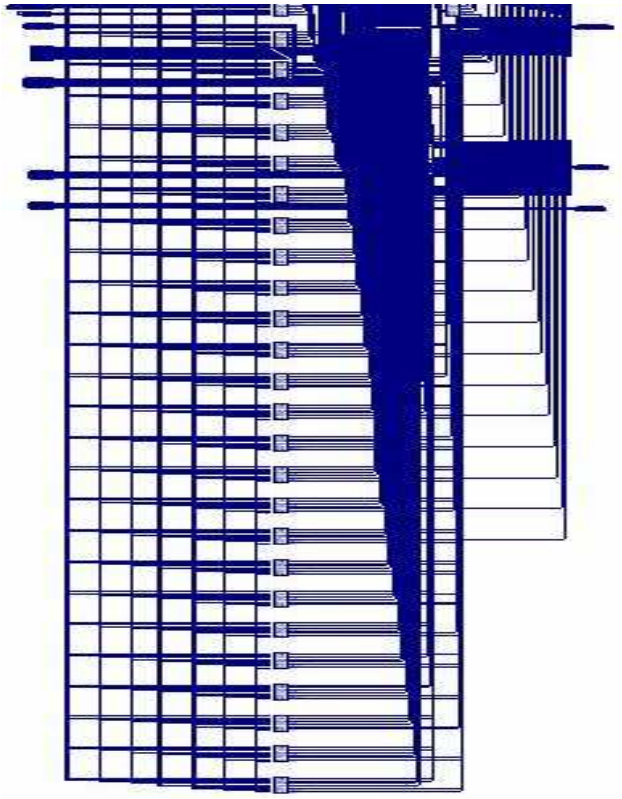


Figure 5.14: Schematic of ACS

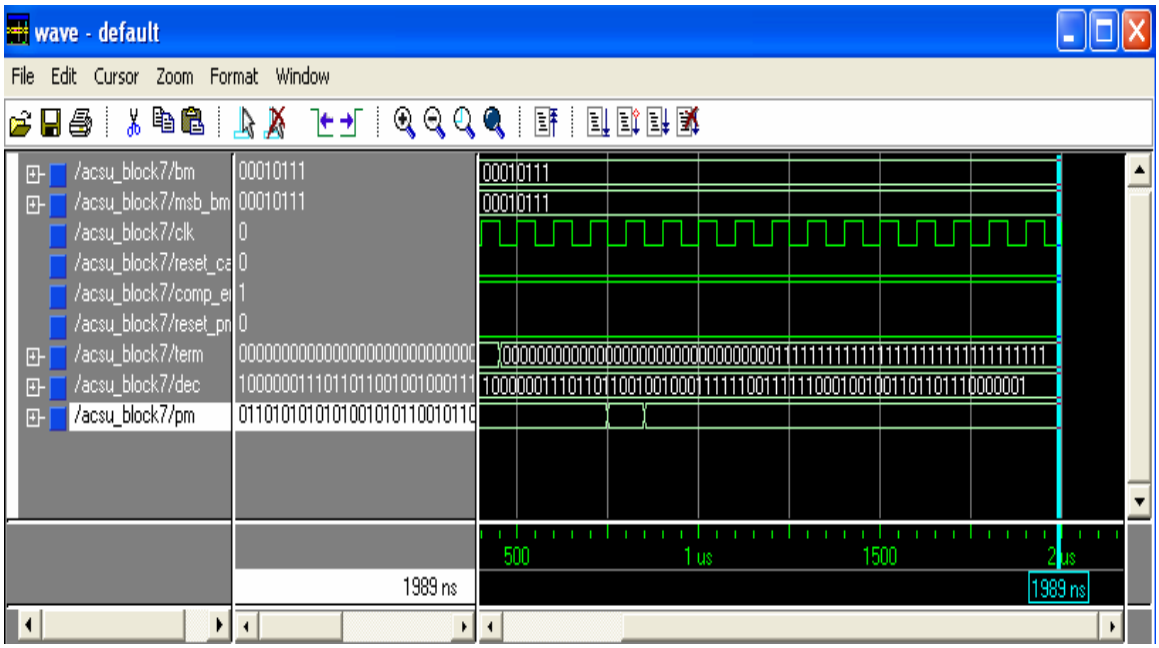


Figure 5.15: Simulations of ACS

5.1.6. SIMULATION A ND SYNTHESIS RESULTS OF ACSTOMUX

The inputs to the pointer module are decision_in, termination_in, pointer, termination_in, clk and reset. The output signals generated are decision_out and termination_out. The inputs decision_in and termination_in are of 64 bits, which are inputs from the ACS unit. The signals decision_out and termination_out are of 64 bits.

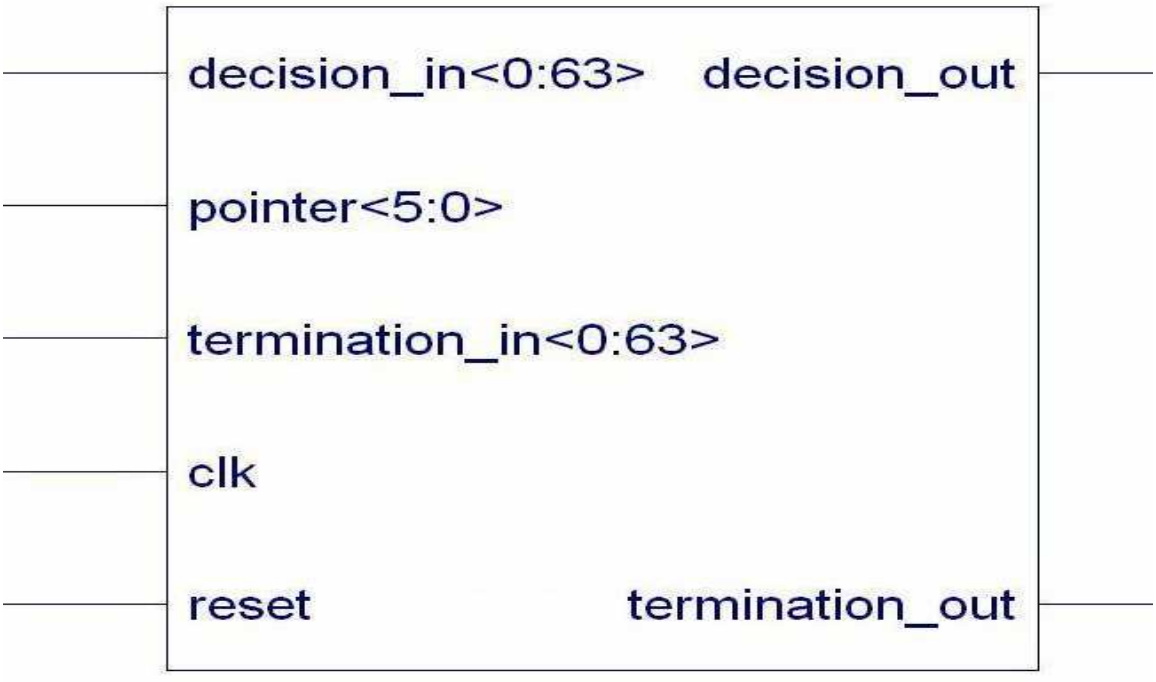


Figure 5.16: RTL of ACSTOMUX

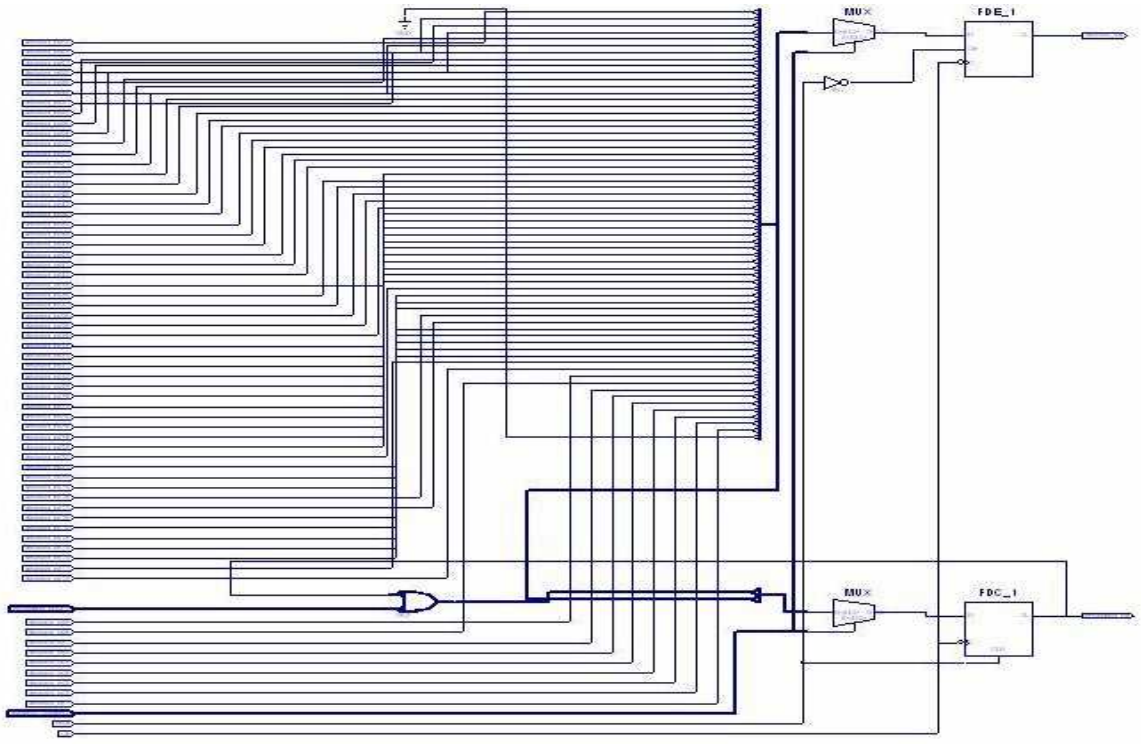


Figure 5.17: Schematic of ACSTOMUX

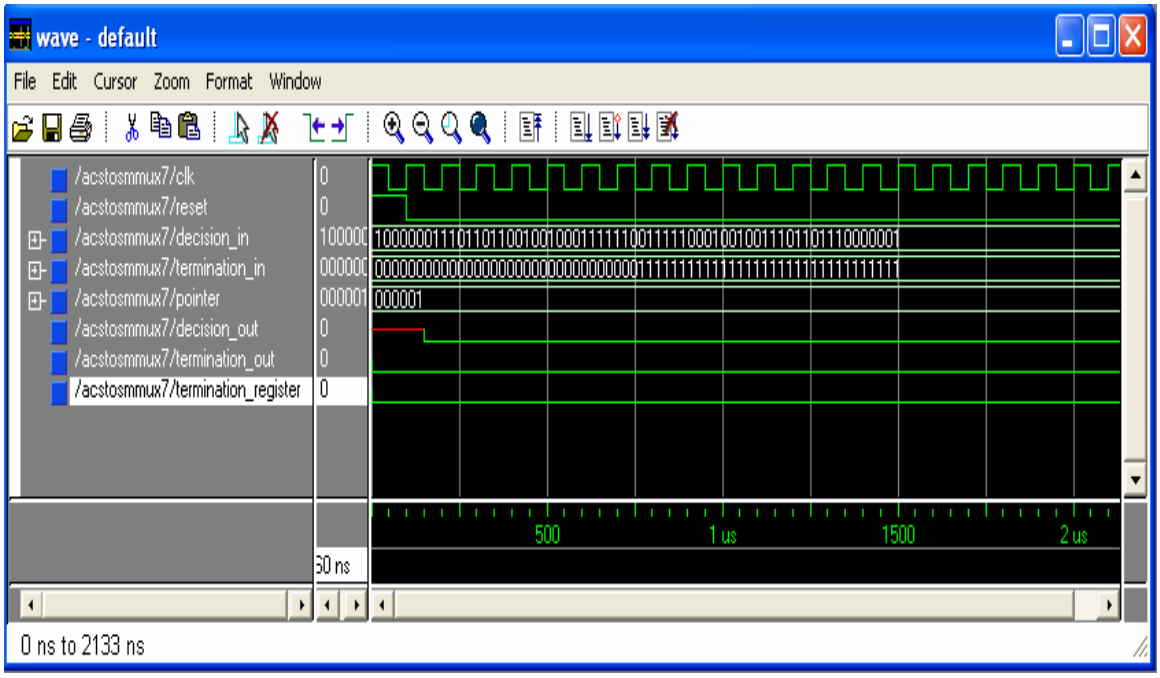


Figure 5.18: Simulations of ACSTOMUX

5.1.7. SIMULATION A ND SYNTHESIS RESULTS OF MSB

The inputs to MSB are reset and clk. The output signal generated from these inputs is MSB which acts input to the pointer module.

Figure 5.19: RTL of MSB

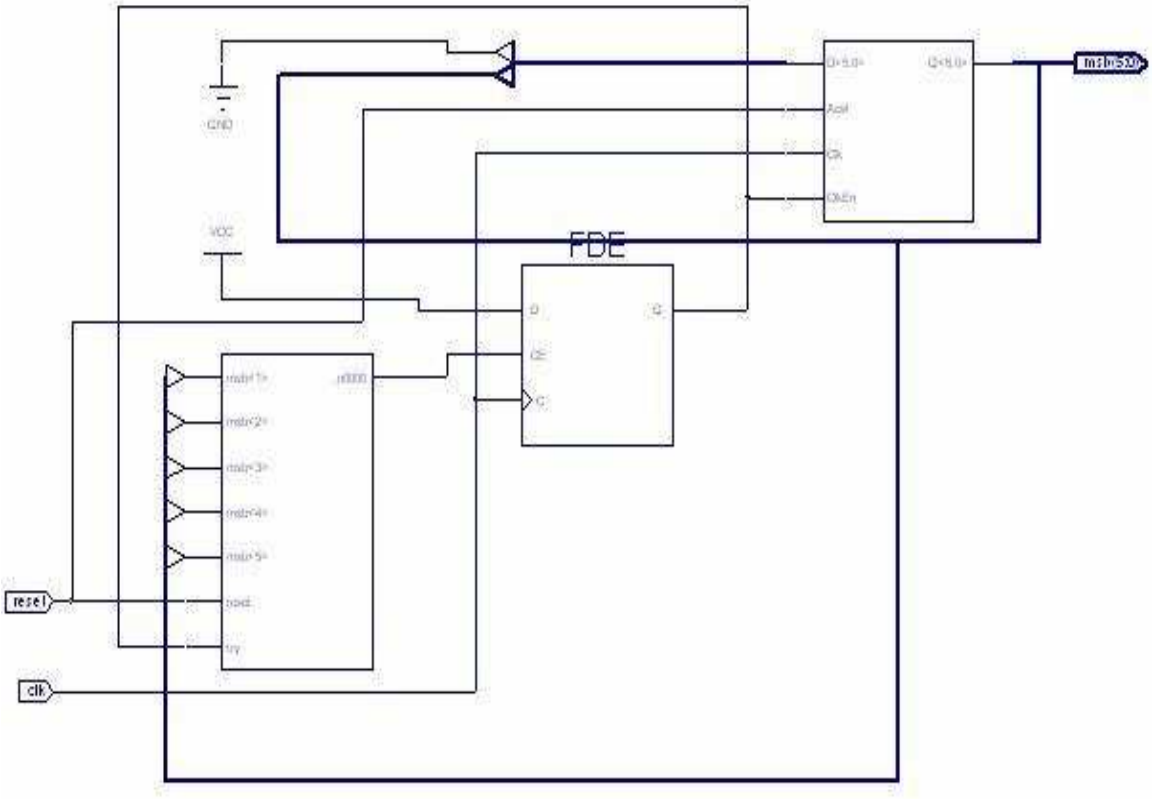


Figure 5.20: Schematic of MSB

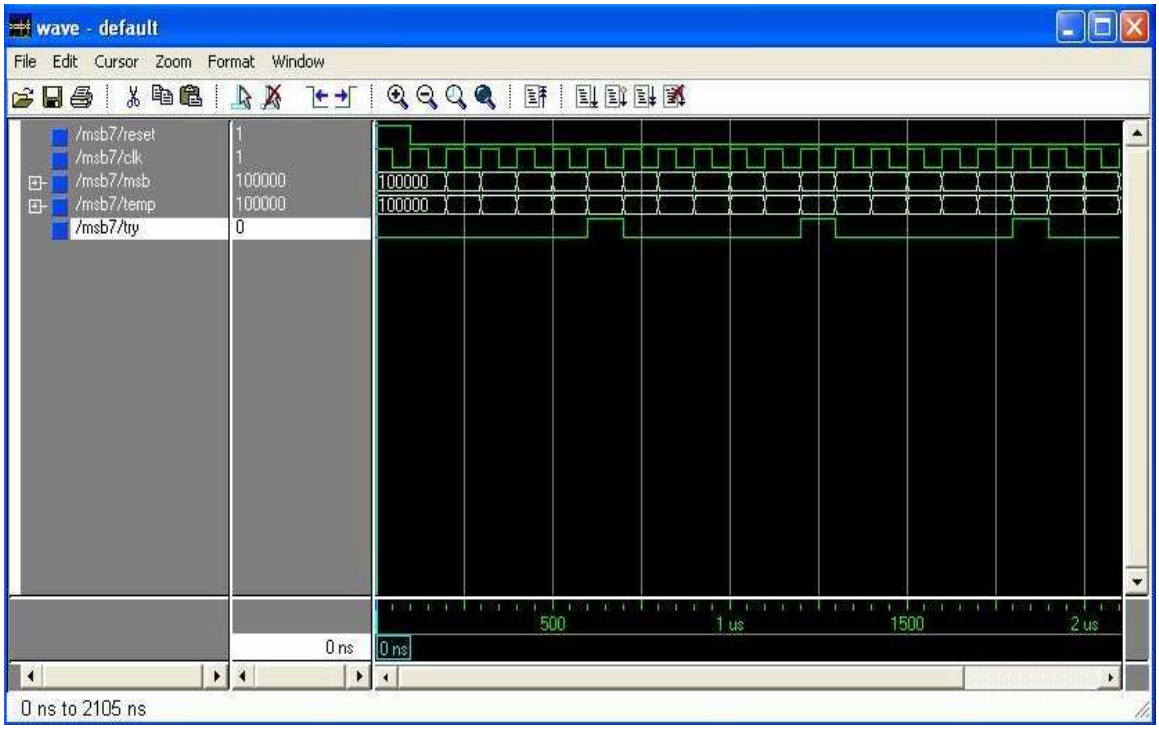


Figure 5.21: Simulations of MSB

6.1.8. SIMULATION A ND SYNTHESIS RESULTS OF POINTER

The inputs to the pointer module are msb, which is of 6 bits, clk, new_bit and reset. The output signal is pointer_out, which is of 6 bits.

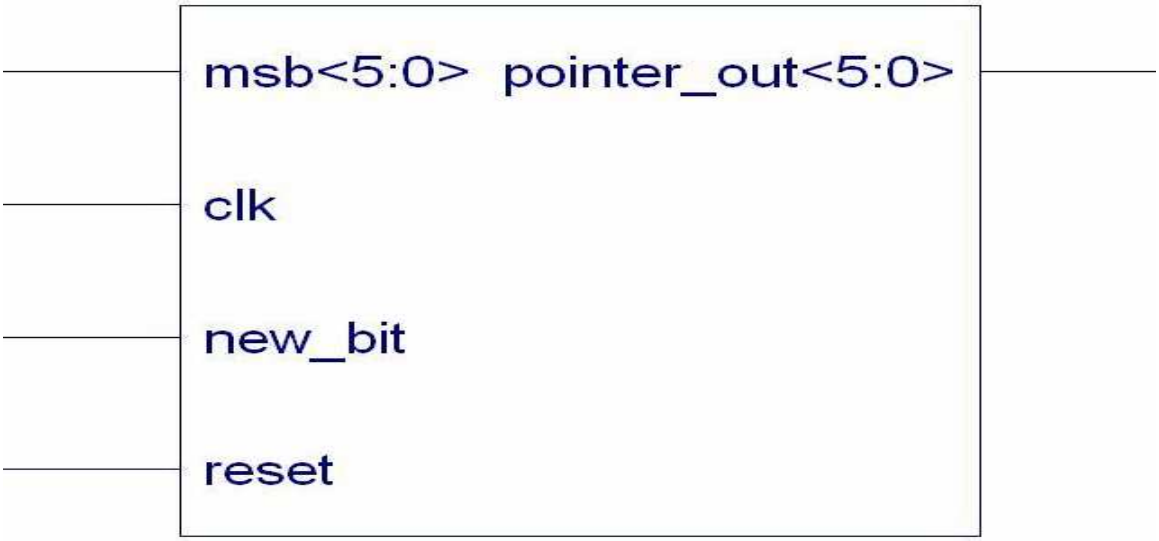


Figure 5.22: RTL of Pointer

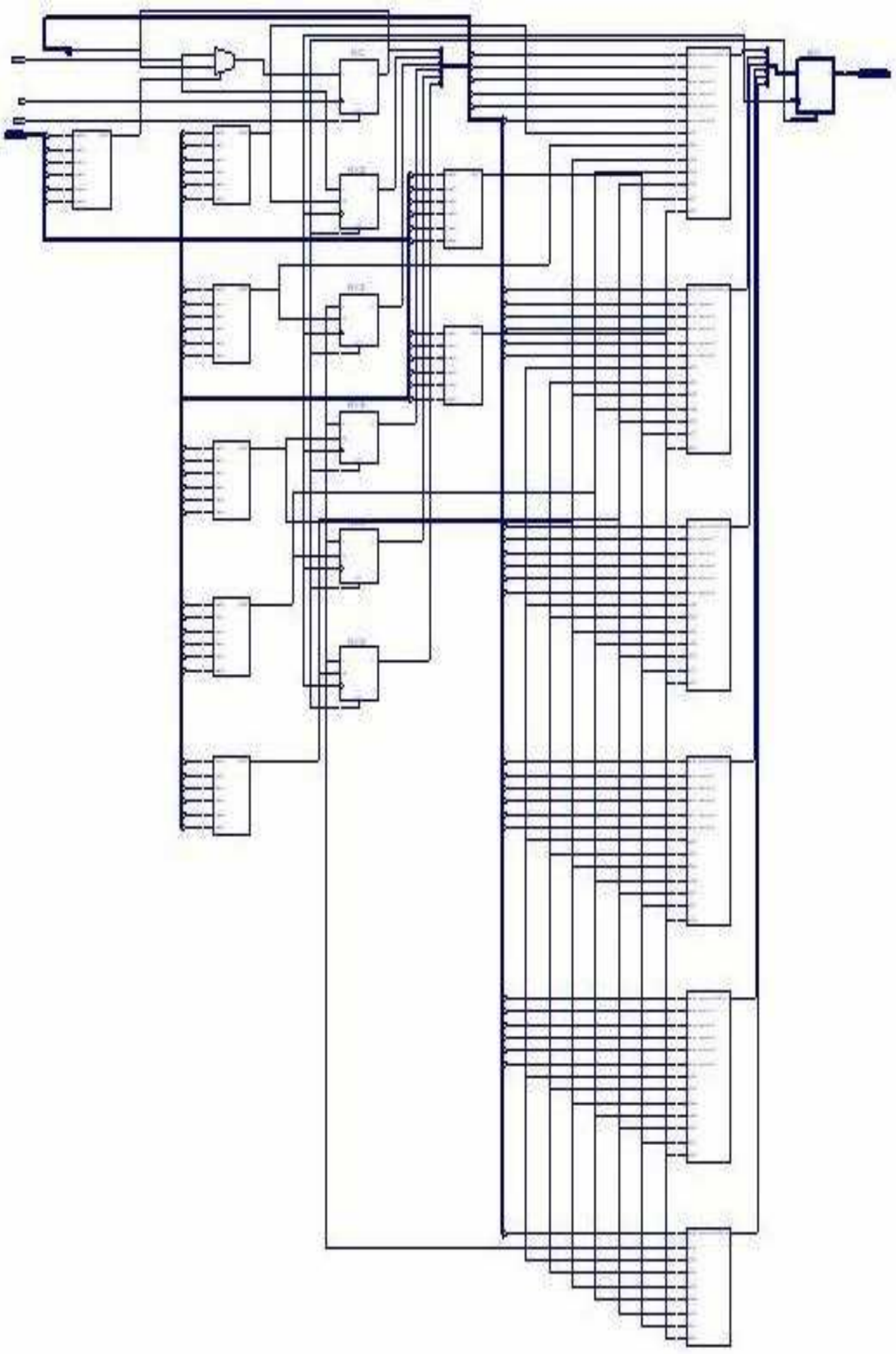


Figure 5.23: Schematic of Pointer

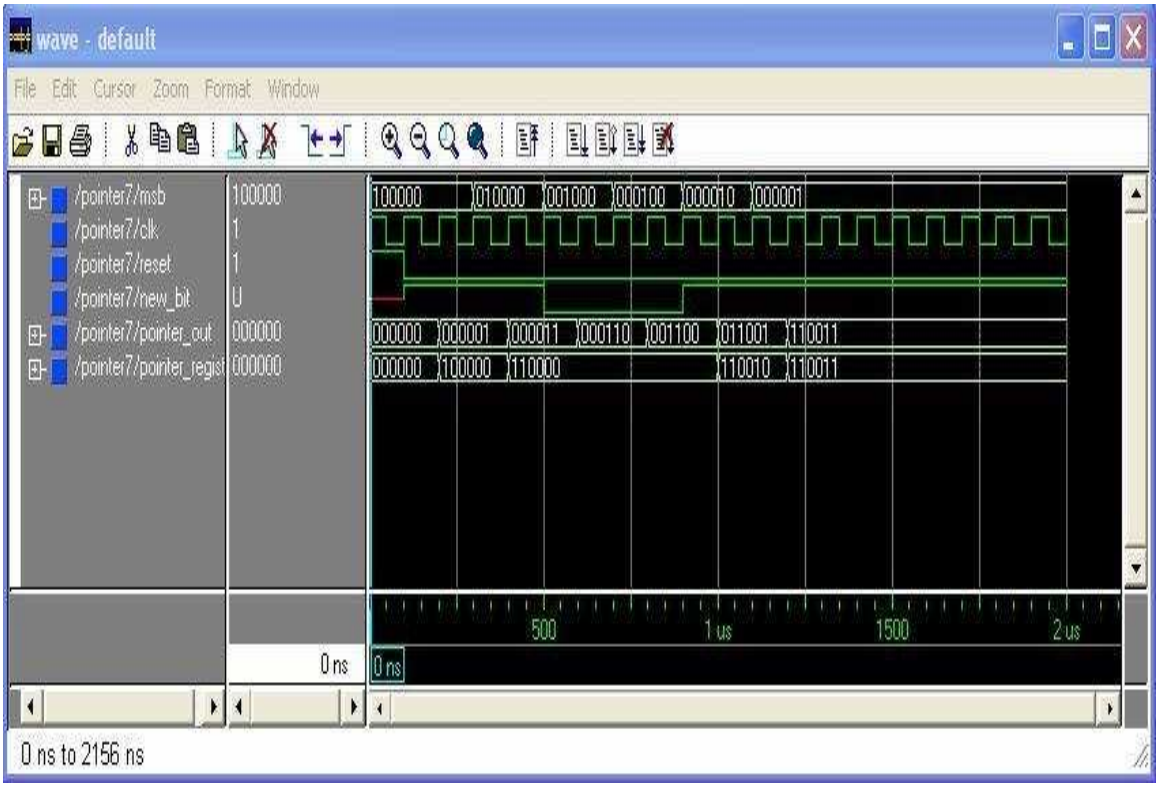


Figure 5.24: Simulations of Pointer

5.1.9. SIMULATION A ND SYNTHESIS RESULTS OF VITERBI MODULE

All the modules are mapped in this module. The structured programming is used in this module. The input signals are clk, start_out. The output signals are data_out and term_out.

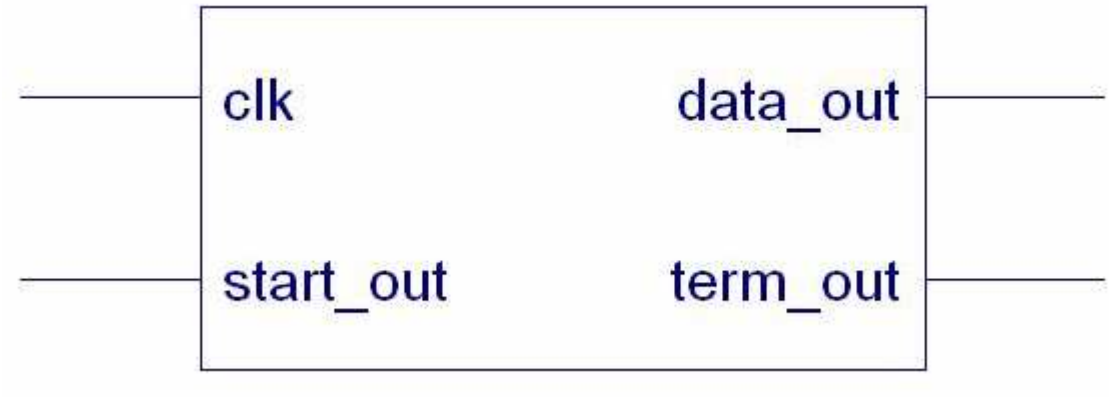


Figure 5.25: RTL of Viterbi

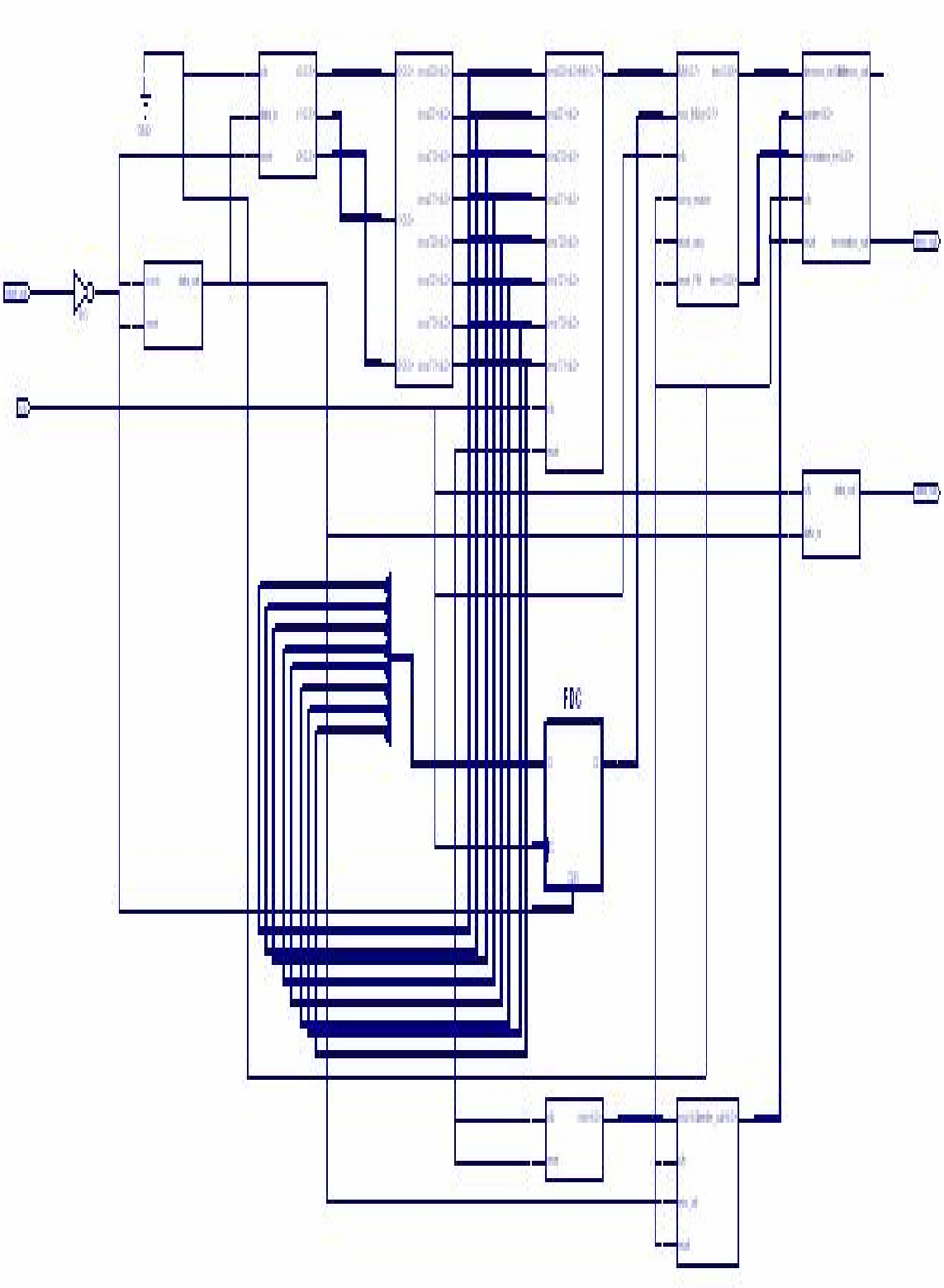


Figure 5.26: Schematic of Viterbi

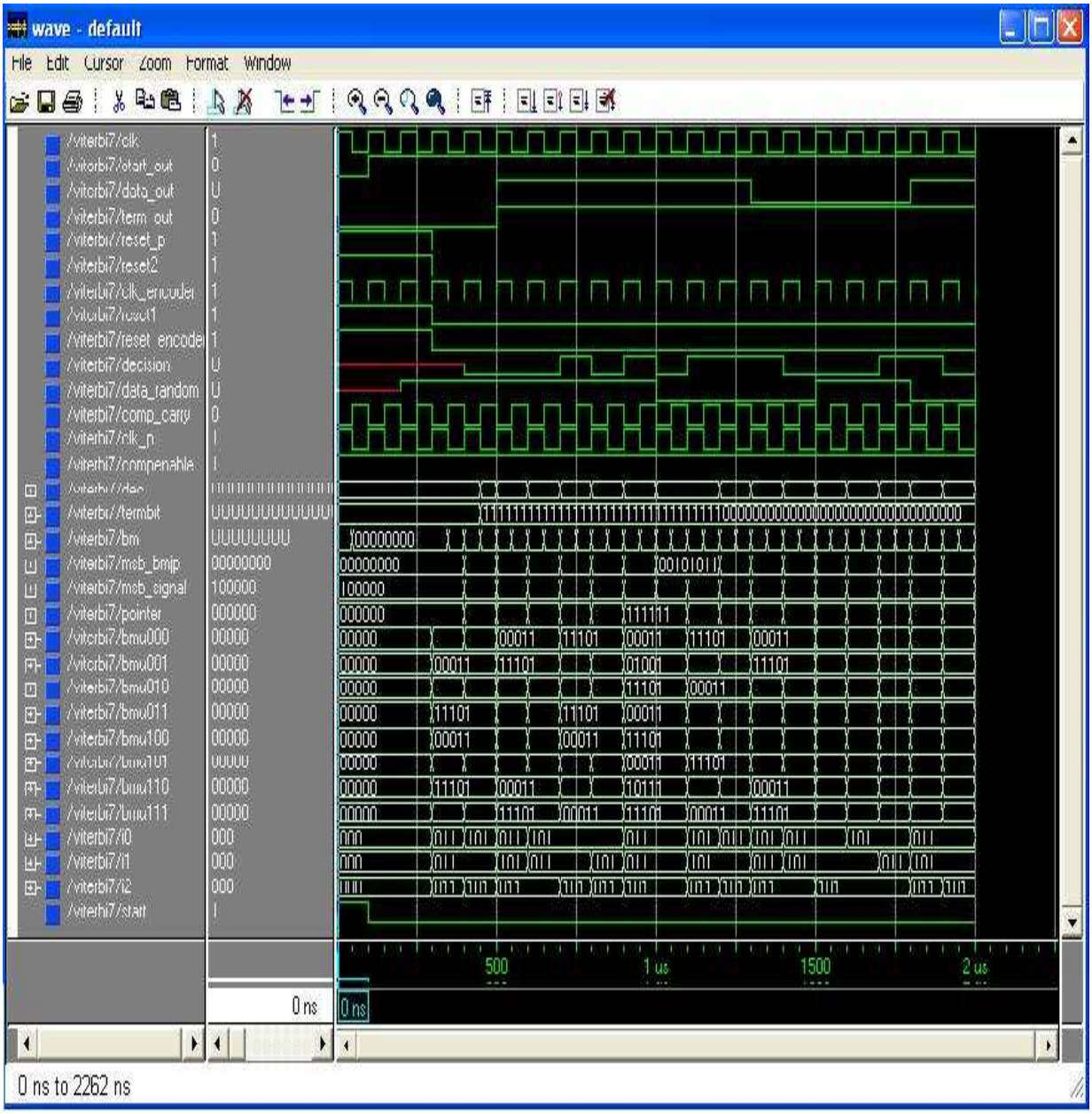


Figure 5.27: Simulations of Viterbi

Final Result: Minimum period: 5.105ns

Maximum Frequency: 195.886MHz

CHAPTER-6
CONCLUSION AND FUTURE SCOPE

Viterbi decoding is the best technique for decoding the convolutional codes. Viterbi decoder with constraint length 7 and code rate 1/3 has been developed. The main consideration is to decrease the power dissipation. The two techniques used for decoding the convolutional codes are the RE method and TB method. TB method is used for larger constraint length but takes more time for decoding. RE method is simplest and fastest but more suitable for small constraint lengths. The RE method with large constraint length (K=7) is implemented with Modified Register Exchange Method. This implementation reduces the power consumption to a large extent. The ACS unit and survivor memory management unit consumes most of the power. Bit serial approach is used for implementation of ACS unit. The amount of interconnections is reduced by using bit serial architecture and the butterfly concept unit which in turn reduces the power consumption. Further, power is reduced by using the modified register exchange method. Modified register exchange method uses the pointer concept. The extra overheads are the registers required for storing the carry bits and the path metrics.

The different modules are designed using VHDL and synthesized using Xilinx Integrated software environment (ISE). The design is simulated using Model Sim. The design implementation is done on Xilinx Spartan 2E xc2s15-6cs144. The maximum operating frequency achieved is 195.886MHz.

The Viterbi decoding is limited to lower constraint lengths. Viterbi decoder with Modified Register Exchange Method can be further investigated for higher constraint lengths.

[1] Vasily P. Pribylov, Alexander I. Plyasunov (2005). "A Convolutional Code Decoder Design Using Viterbi Algorithm with Register Exchange History Unit". SIBCON. IEEE.

[2] John G. Proakis (2001). "Digital Communication". McGraw Hill, Singapore. pp 502-507, 471-475

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

[4] Irwin M. Jacobs (1974). "Practical Applications of Coding". IEEE. pp 305-310.

[5] Xun Liu Marios C. Papaefthymiou. "Design of a High-Throughput Low-Power IS95 Viterbi Decoder"**.** Advanced Computer Architecture Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan.

[6] Tommy Oberg (2001). "Modulation Detection and Coding". Wiley and Sons. pp 81- 86.

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

[8] Stefan Bitterlich and Heinrich Meyr (1993). "Efficient Scalable Architectures for Viterbi Decoders". Aachen University of Technology, Templergraben , Germany. pp 89-100.

[9] Sang-Cheon Kim, Je-Hyuk Ryu, and Jun-Dong Cho. “Low Power High-Rate Viterbi Decoder Employing the SST (Scarce State Transition) Scheme and Radix-4 Trellis” Department of Electrical and Computer Engineering, Sungkyunkwan University.

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

[11] I. Bogdan, M. Munteanu, P. A. Ivey, N. L. Seed, N. Powell. “Power Reduction Techniques for a Viterbi Decoder Implementation”. Electronic Systems Group, University of Sheffield, Mappin Street, Sheffield S1 3EA, UK.

[12] Samirkumar Ranpara and Dong Sam Ha. “Low-Power Viterbi Decoder Design for Wireless Communications Applications”.

[13] Inyup Kang, Member IEEE and Alan N. Willson (1998). “Low Power Viterbi Decoder for CDMA Mobile Terminal”. IEEE Journal of Solid State Circuits. IEEE. Vol 33. pp-473-481.

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

[15] Jun Tang and Keshab K. Parhi (2005). “V.D for High Speed Ultra Wideband Communication Systems”. IEEEE. Vol. 37.

[16] Dalia A.El-Dib and Mohammed I-Elmasry. (2005). “Memoryless Viterbi Decoder”, IEEE. Vol. 52. pp 826-830.

[17] Chaiwat Keawsai, Keattisak Sripimanwat,Attasit Lasakul. “Modified register exchange method of Viterbi decoder for 3GPP Mobile System”.

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

[19] Christian Schuler and GMD IOKH. “Code Generation Tools for hardware implementation of FEC Circuits”.

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

[21] John Davis, Andrew Lin, Njuguna Njoroge, AyodeleThomas (2002). “The Viterbi algorithm as stream application”.

(1) Pushpinder kaur, Charu Gupta, Sanjay Sharma and Ajaypal Singh "FPGA Based FIR Implementation", published in ECCS 2006 organized by Electronics and Communication Engineering Department at Thapar Institute of Engineering and Technology, Patiala, Page(s): 336-338, held on 9^{th}-10^{th} Feb, 2006.

(2) Charu Gupta, Pushpinder Kaur and Sanjay Sharma," Free Space Optics" published in ECCS 2006 organized by Electronics and Communication Engineering Department at Thapar Institute of Engineering and Technology, Patiala, Page(s): 310-313, held on 9^{th}-10^{th} Feb, 2006.

(3) Charu Gupta, Pushpinder Kaur and Sanjay Sharma," Character Recognition using Neural Networks" published in ECCS 2006 organized by Electronics and Communication Engineering Department at Thapar Institute of Engineering and Technology, Patiala, Page(s): 184-186, held on 9^{th}-10^{th} Feb, 2006.

(4) Paper accepted in National Conference on Low Power Viterbi Decoder at Chitkara Institute of Engg. and Technology, Rajpura, Punjab.

Equation 1