Implementation of Viterbi Decoder for Error Detection and correction

[1]M Likhitha, [2]Bindu Vaishnavi Y, [3]TVN Srikar, [4]Y Sidhanth

[1]Student, [2] Student, [3] Student, [4] Student
Electronics and Communication Engineering,
Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology, Hyderabad, India

***Abstract*: Data transfer is critical in communication networks. On the Transmitter side, the messages are encoded, and on the Receiver side, they are decoded. Due to many interruptions in the communication connection, data may become distorted during message transmission. As a result, the decoder tool must also have the ability to correct any errors that may occur. This paper deals with designing and implementing a convolutional encoder and Viterbi decoder which play a major role in the building blocks of digital communication. Because of its error detection and correction capabilities, the Viterbi algorithm has a wide range of applications. Viterbi decoding is a strong forward error correcting technique. With a specific path management unit, the Viterbi Decoder is designed for higher decoding speed and smaller routing space. Verilog HDL is used to build the system. Xilinx 9.1 is used for simulation and Altera Quartus is used for synthesis**.

***Index Terms*: Viterbi Algorithm, Trellis diagram, Convolutional encoder, Branch metric unit, Traceback unit, Hamming distance.**

I. INTRODUCTION

Error correction codes are essential to make sure that the information content of the message if altered could be corrected without having to add much overhead to the system. The convolution coding technique, on the other hand, works on a bit-by-bit basis, that is, they operate on a stream of bits of any length. When using block coding, each packet or frame is considered independent of the others, whereas convolution codes operate on a continuous stream of data in real-time. It then performs an encoding operation to create a larger codeword that will be the actual data word. carrying information in an encrypted pattern. Convolutional encoding is frequently employed to correct random errors.

The Viterbi decoder performs maximum likelihood decoding using the trellis diagram, which simplifies the computation. The Viterbi algorithm has become the most fundamental algorithm in decoding techniques, with applications in CDMA, GSM, satellite systems, speech recognition, enhancement, and synthesis, as well as many other data decoding technologies. In addition, by segmenting the code words into words, a Viterbi decoding method can be applied to a larger codeword. Forward error correction (FEC) techniques are used in the transmitter to encode the data stream and in the receiver to detect and correct errors in the data stream, thereby lowering the bit error rate (BER) and improving performance.

The primary goal of this study is to perform a functional analysis of the error control coding technique, i.e., convolution encoding, in terms of device utilization and power. The functional verification is carried out through simulation with Verilog HDL in the Xilinx.

Convolutional encoding with Viterbi decoding is a powerful FEC technique that is well suited to a channel where the transmitted signal is primarily corrupted by AWGN. It operates on a data stream and has a memory that encodes previous bits. It is simple, performs well, and has a low implementation cost. The convolutional encoder is padded with zeros to a continuous stream of input data by using a linear shift register.

II. METHODOLOGY

A. CONVOLUTIONAL ENCODER

A convolutional encoder accepts a sequence of message symbols and produces a sequence of code symbols. Its computations depend not only on the current set of input symbols but on some of the previous input symbols as well. In practice, convolutional codes operate on a block at a time and so like block codes, have intra-block memory, and possibly no inter-block memory.

Convolutional Encoder: A convolutional encoder is made of a fixed number of shift registers. Each input bit enters a shift register and the output of the encoder is derived by combining the bits in the shift register. The number of output bits depends on the number of modulo 2-adders used with the shift registers.

Encoder Parameters: Convolutional codes are commonly specified by the three parameters
(n, k, m), where, n = number of output bits
k = number of input bits and,
m = number of memory registers.

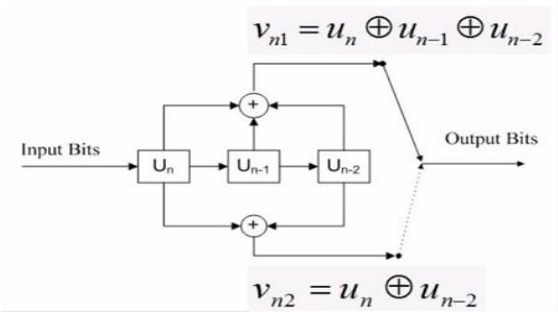

Fig 1: Convolutional encoder block diagram

Convolutional codes can also be specified by the parameters (n, k, L) where L is known as the constraint length of the code and is defined as the number of bits in the encoder memory that affects the generation of the n output bits. The convolutional codes discussed here will be referred to as (n, k, L) and not as (n, k, m) codes.

There are four registers in the encoder. At the k-th time, input symbol m_k of sequence, M = (m_1, m_2, m_3,...) will make two output symbols C^i_k C^i_k , i=1,2. with

$$C^i_k \ (i=1) = m_k \oplus C \oplus D$$
$$C^i_k \ (I=2) = m_k \oplus A \oplus C \oplus D$$

The output sequence is c = (C^i_1 C^i_1, C^i_2 C^i_2, C^i_3 C^i_3, …) where(i=1,2). Each code segment has two bits. The quantity k / n called the code rate is a measure of the bandwidth efficiency of the code. Commonly k and n parameters range from 1 to 8, m from 2 to 10, and the code rate from 1/8 to 7/8 except for deep-space applications where code rates as low as 1/100 or even longer can be employed.

B. VITERBI DECODER

The viterbi decoder is most commonly used to resolve convolution codes. It has a special property of reducing the bits by half which reduces the redundancy in code; hence it is the most efficient decoder. It converts the encoded data bits to the original form. A Viterbi decoder uses the Viterbi algorithm for decoding a bit stream that has been encoded using a convolution. It is mostly used for constraint length (k), k <=10.

This algorithm is broken down into 3 different phases:

1. Branch Metric unit
2. Add compare select
3. Traceback unit

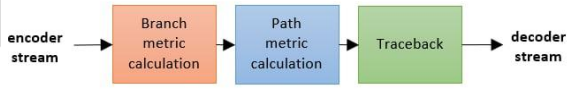

Fig 2: Viterbi decoder block diagram

1.Branch Metric unit

Due to the normal unit and original unit of the message bits of the branch metric unit, the major unit of Hamming distance. The hamming distance was used in their separation technique for the branch metric unit. From the input data, the BM unit is utilized to calculate branch metrics for all trellis branches. As a branch metric, we use the absolute difference as a metric. These branch metrics are regarded as equating the branch weights.

2. Add Compare Select

To generate the trellis of the decision, select the add compare option and process the sub modules on the path metric unit. When the path contains two stages, measurements are taken to determine the present state. To choose the transmitted data for the add compare choose and loop operation on the stage, use the path metric. In the process of adding oneself to characterize a specific path. When the shortest path number is at the coding bits, the code trellis of the shortest path number is used.

3. Traceback Unit

Results of these are written to the memory of the traceback unit. It Traces back from the end of any survivor's path to the beginning. The decoded data is saved in the survivor path to add and compare select units. When a choice is made to allow current data to flow into the ACS unit's register level path. To produce the input to the ACS output at the next level of the survival unit. When a lot of people employ the data exchange mechanism. The register level exchange takes longer to convert.

C. TRELLIS DIAGRAM

Trellis Diagrams are formed up of a matrix of nodes. Each node in the status diagram reflects a different status. Each node's column represents all of the conceivable states at any given time. For example, at the state machine's initial state (t = 0), the first column (the leftmost column) displays all potential states. When t = 1, the second column represents all of the potential statuses. When t = 2 and so on, the third column represents all of the potential states.

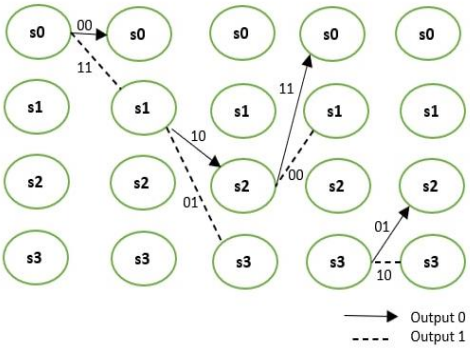

Fig 3: Trellis diagram

Each node may have two options for progressing to the next status. When the input bit is 0, one path is the transition path, and when the input bit is 1, the other path is the transition path.

III. PROPOSED METHODOLOGY

This proposed Viterbi decoder mainly functions with the concept of main machine and sub machine. The top module consists of 6 modules. Namely: Viterbi main machine, Viterbi sub machine 1, Viterbi sub machine 2, Comparator, Multiplexer, Temporary registers.

1. Viterbi main machine: This Viterbi main machine consists of 4 states in it. Inputs to this main machine are clock, reset, State final (output of Multiplexer), and input from Temporary register 2. Outputs to this are output bit, state, and enable.
2. Viterbi sub machine1: This Viterbi main machine consists of 2 states (S0, S2) in it and works only when enable is high. Inputs to this main machine are clock, reset, and input from Temporary register 1. Outputs to this state, and min path1.
3. Viterbi sub machine2: This Viterbi main machine consists of 2 states (S1, S3) in it and works only when enable is high. Inputs to this main machine are clock, reset, and input from Temporary register 1. Outputs to this state, and min path2.
4. Comparator: The main purpose of the comparator module is to compare the min path1 and min path2 and send the output signal to the Multiplexer.
5. Multiplexer: The selection line in the multiplexer is the output signal of the comparator and the input is state representation 1 and state representation 2. output is state final.
6. Temporary registers: In this whole Viterbi module totally 2 temporary registers are used which are temp1 and temp2. Input to temp 1 is the output of the convolutional encoder and output is given to temp2 and Viterbi sub machine 1 & Viterbi sub machine2. The input of temp2 is from temp1 and the output is given to the Viterbi main machine.

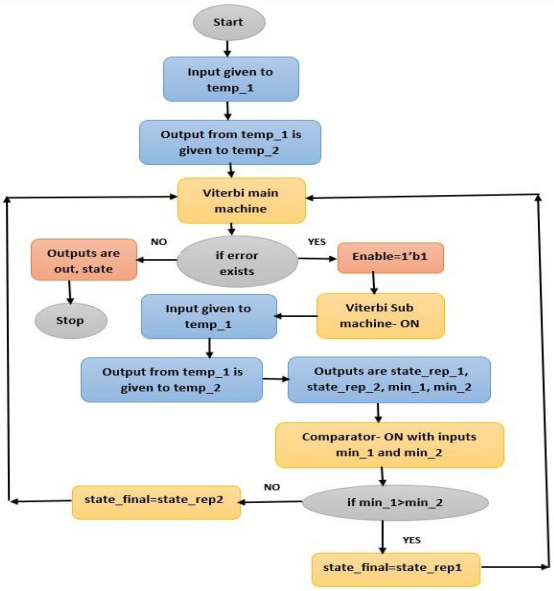

Fig 4: Flow Chart

Consider a sequence without an error. Example: 1101.The bits are given to temp1, at the positive edge of clock. The bit from temp1 is sent to temp2. The temp2 is connected to the viterbi main machine. The present state in viterbi main machine is s0 and input 2 bits are 11. Since there is no error in viterbi main machine so enable is 0 and output is 1, state changes from s0 to s1.

Next 2 bits are 01 and viterbi present state is s1, as there is no error enable is 0 and output is 1, state changes to s3.

Consider another sequence with an error Example: 0110 Consider an example with sequence 0110 as the input to the Viterbi decoder. The bits are given to temp_1 at the positive edge of the clock. The bits from temp_1 is shifted to temp_2. The data in temp_2 is sent to the Viterbi main machine where the present state is s0. The bits 01 in the main machine having error, so the enable will become 1. So, both sub machines will get on and take the next bits 10.sub machine 1 contains states (S0, S2) and sub machine 2 contains states (S1, S3). As input is 10 output of sub machine 1 will be min_path1 is 1 state_representation1 is S0 and output of sub machine 2 will be min_path2 is 0 state_representation1 is S1.min_path1 and min_path2 which are outputs of both sub machines are given to comparator. If min_path1>min_path2 then the output will be 1 or else it will be 0. The output of the comparator is used as a selection line for the multiplexer and inputs are state_representation1 and state_representation2. If the selection line is 1 then state_final will be state_representation2 or else state_final will be state_representation1. this state_final is given to the main Viterbi machine and from there with that state_final main Viterbi machine will continue.

IV. SIMULATION RESULT

For simulation, Xilinx 9.1 is used and for synthesis, Altera Quartus is used.

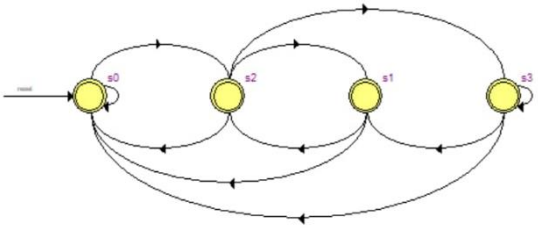

Fig 5: State diagram of convolutional encoder

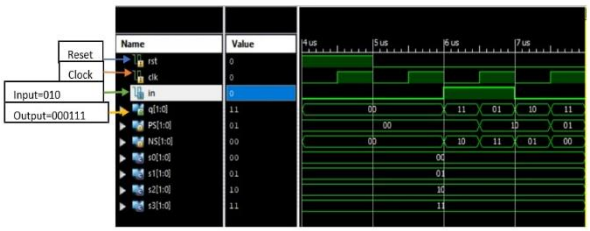

Fig 6: Simulation results of convolutional encoder

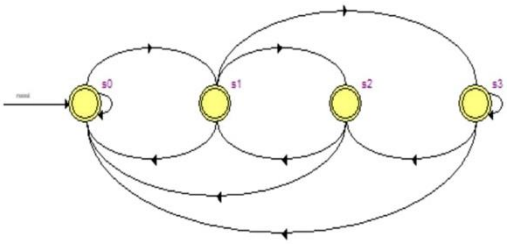

Fig 7: State diagram of Viterbi decoder

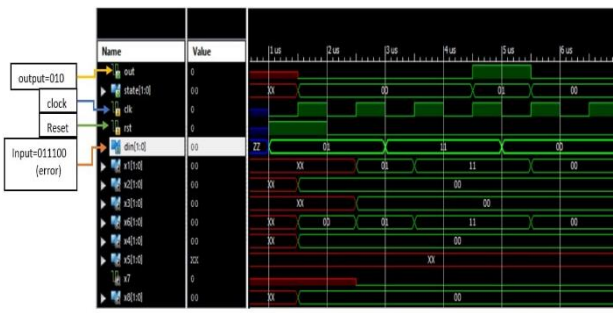

Fig 8: Simulation results of Viterbi decoder

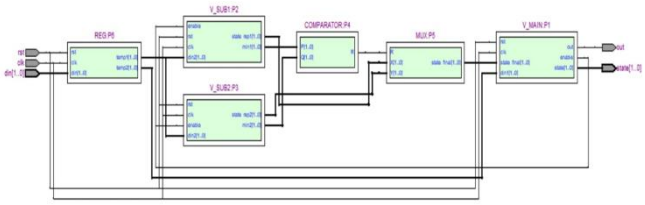

Fig 9: RTL view of Viterbi decoder

V. CONCLUSION

In this paper, we implemented an optimized solution for a widely used error detection and correction Viterbi decoder using the concepts of main machine and sub machine. Architecture modifications are done for better performance and exploration of error detection and correction.

VI. FUTURE SCOPE

In this proposed at the input side, we can add the SIPO register and at the output side, PISO register can be added in order to reduce the propagation time and to increase the performance.

VII. REFERENCES

[1] A. Thakur and M. K. Chattopadhyay, "Design and Implementation of Viterbi Decoder Using VHDL," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 331, no. 1, 2018, doi: 10.1088/1757-899X/331/1/012009.

[2] S. Mishra and R. R. Tripathi, "VDHL Implementation of Viterbi Algorithm for Decoding of Convolutional Code," *Proc. - 2015 Int. Conf. Comput. Intell. Commun. Networks, CICN 2015*, no. 1111001, pp. 1367–1370, 2016, doi: 10.1109/CICN.2015.265.

[3] K. S. Kumar and M. V. H. B. Murthy, "FPGA Implementation of Viterbi Algorithm for Decoding of Convolution Codes," *IOSR J. VLSI Signal Process.*, vol. 4, no. 5, pp. 46–53, 2014, doi: 10.9790/4200-04514653.

[4] S. W. Shaker, S. H. Elramly, and K. A. Shehata, "FPGA implementation of a reconfigurable Viterbi decoder for WiMAX receiver," *Proc. Int. Conf. Microelectron. ICM*, pp. 264–267, 2009, doi: 10.1109/ICM.2009.5418636.

[5] K. Cholan, "Design and implementation of low power high speed viterbi decoder," *Procedia Eng.*, vol. 30, no. 2011, pp. 61–68, 2012, doi: 10.1016/j.proeng.2012.01.834.

[6] A. Khare, M. Saxena, and J. Patel, "FPGA Based Efficient Implementation of Viterbi Decoder," *Int. J. Eng. Adv. Technol.*, no. 1, pp. 2249–8958, 2011.

[7] V. Kavinilavu, S. Salivahanan, V. S. K. Bhaaskaran, S. Sakthikumaran, B. Brindha, and C. Vinoth, "Implementation of convolutional encoder and Viterbi decoder using Verilog HDL," *ICECT 2011 - 2011 3rd Int. Conf. Electron. Comput. Technol.*, vol. 1, pp. 297–300, 2011, doi: 10.1109/ICECTECH.2011.5941609.

[8] V. P. Patil, D. G. Chougule, and R. R. Naik, "Viterbi Algorithm for error detection and correction," pp. 60–65, 1967, [Online]. Available: www.iosrjournals.org

[9] K. M. Abubeker, A. R. Bushara, and S. Backer, "Maximum likelihood de coding of convolutional codes using viterbi algorithm with improved error correction capability," *2013 IEEE Conf. Inf. Commun. Technol. ICT 2013*, no. Ict, pp. 161–164, 2013, doi: 10.1109/CICT.2013.6558082.