International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Design and Verification of Storage Efficient Viterbi Decoder

Divyashree V[1], Ranjeetha M V[2] , K. V. Prasad[3]

[1]PG Student
[2]Assistant Professor
[3]Professor and Head of the Department
[1,2,3]Dept. of Electronics and Communication Engineering, Bangalore Institute of Technology, Karnataka, India

---***---

Abstract - *Data transmission over the wireless channel is adversely affected by attenuation, distortion, interference, and noise, which hampers the ability of the receiver to correctly receive the transmitted message signal. Thus error detection and correction methods are implemented to mitigate these effects. Convolution encoder is one such channel encoding technique used at the transmitter end for deep space and wireless communication whereas at the receiver end the Viterbi decoder decodes the encoded data. Viterbi algorithm is based on principles of maximum likelihood where the optimal trellis path is identified that is followed at the encoder using cumulative hamming distance.*

This paper focuses on improving the memory and area utilized by the Viterbi decoder and comparison between conventional and proposed decoding method. The soft IP (Intellectual Property) is developed as per ASIC and FPGA design methodology. This had been modeled using Verilog HDL, Functionally Verified using incisive simulator for ASIC flow and Questa Simulator for FPGA flow, synthesized in genus and Precision RTL with a clock frequency of 100 MHz, for ZYNQ-7Z020CLG400 Vivado device. Finally Equivalence Check was performed using FormalPro.

***Key Words*: Convolution Encoder, Viterbi Decoder, ASIC Flow, FPGA Flow.**

1.INTRODUCTION

Digital communication is a type of communication where the information is encoded digitally as discrete signals and transferred to the recipients through wireless media. Digital signals are less affected by distortion, noise, and interference and are more reliable[1].

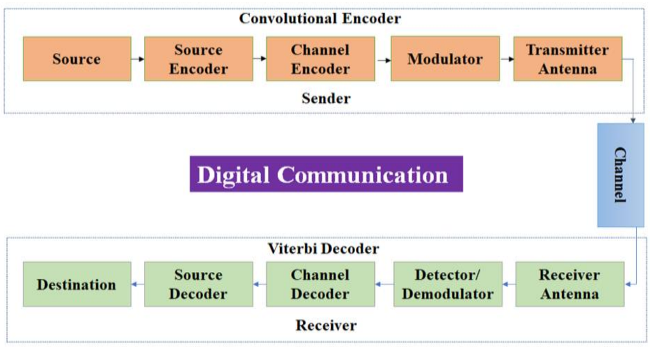

Fig 1: Digital Communication System

Fig 1 represents the basic building blocks of an digital communication system. Input information sequences are compressed using source encoder with less redundancy bit. The data is passed through the channel encoder which introduces redundancy bits on the binary information sequence that is used by the channel decoder at the receiver to eliminate the effects of noise and interference encountered while the signal is being transmitted through the communication channel. Hence, the redundancy bit along with the information message helps in increasing the reliability of the received data and also improves the fidelity of the received signal[2]. Thus, the channel encoder aids the receiver in decoding the desired information sequence. Some of the popular channel encoders are Low-Density Parity Check (LDPC) codes, Turbo codes, Convolution codes, and Reed-Solomon codes and there are two primary decoding strategies. They are based on a maximum a posteriori (MAP) algorithm and a soft-output Viterbi algorithm (SOVA).

2. CONVOLUTION ENCODER

A convolutional code is used to encode a stream of information symbols into a stream of codeword symbols. The general non-systematic convolutional encoder is an application of k parallel serial-in parallel-out variable length shift registers.

A Convolution Encoder (n, k, m) where integer k is the number of data bits, integer n is the total number of codeword out of the decoder. The ratio of k/n is called the rate of the code. The integer m is a parameter know as Constraint length it represents the number of the k-tuples stage in the encoding shift register. Convolution Encoder accepts a 'k' bit input stream of a message and generates 'n' bit encoded output streams to be transmitted[3]. Table 1 describes the design parameter for the given Example.

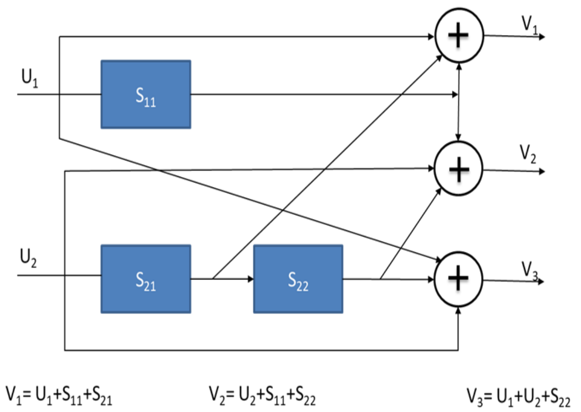

Fig 2: Connection Representation of (3,2,2) Convolution Encoder

The connection representation of the encoder is to specify a set of 'n' connection vectors, one for each of the n modulo-2 adders.

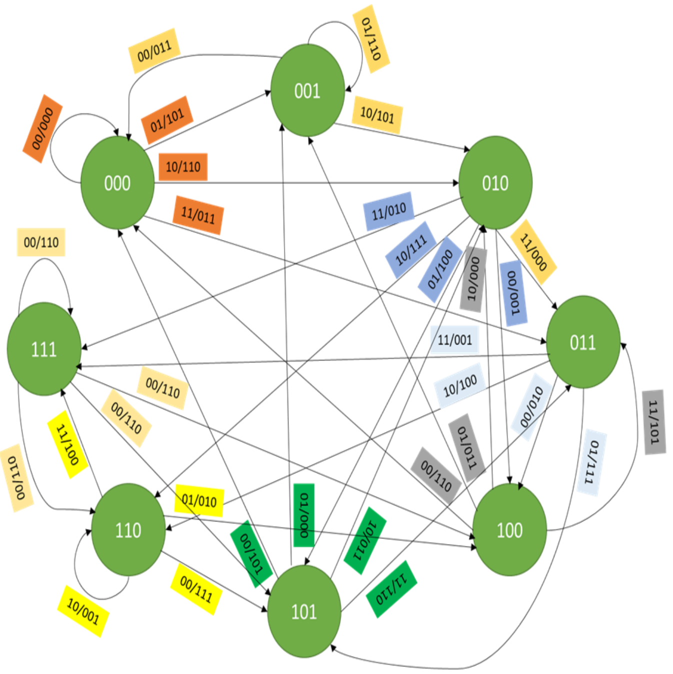

Fig 3: FSM Representation of Convolution Encoder

Table 1: Design Parameters for Convolution Encoder

Parameters	Description	In Given Example
N	Number of output bits	3
K	Number of Input bits	2
m	Number of Memory Registers	2
L	Constraint Length	2
k/n	Code rate	2/3

3. VITERBI DECODER

Several techniques are used to decode convolutional codes, the Viterbi algorithm being one technique. Among all the techniques, the Viterbi algorithm is of major importance because it is based on Maximum Likelihood (ML) technique[4].

In practice, there are a many distinct Viterbi decoder architecture implementations. Fig 4 shows the practical implementation of the Viterbi decoder

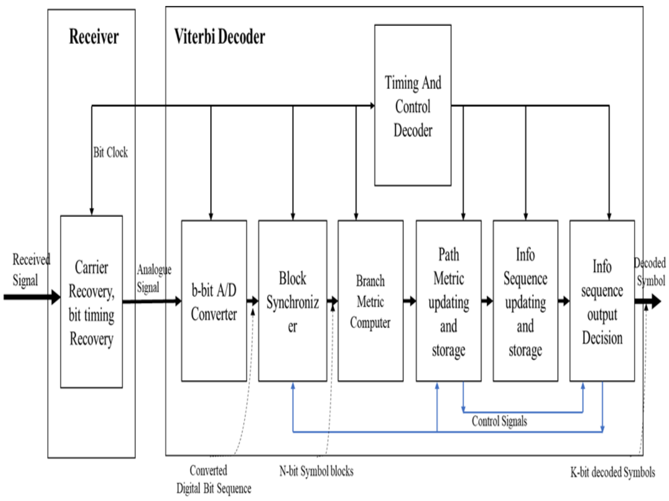

Fig 4: Structure of a Practical Viterbi Decoder.

Quantization (*b*-bit A/D Converter)

The b-bit A/D Converter can be configured to convert the incoming analogy signal to a *b*-bit digital sequence.

Block Synchronization

The block synchronizer can also help resolve polarity ambiguity in the receiver Each point assigns ones and zeros to the received signals in a different manner.

Branch Metric Computer

Branch metric computer is typically based on a look-up table containing the various bit metrics. The computer looks up 'n' bit metrics associated with each branch and sums them to obtain the branch metric[5].

Path Metric Updating and Storage

The path metric updating and storage unit takes the computed branch metrics and computes the partial path metrics at each node in the trellis. These functions are performed most efficiently by breaking the trellis up into many of identical elements[6].

For example, the trellis diagram for (3,2,2) convolutional codes can be broken up into elements containing a pair of origin and destination states and 16 interconnecting branches, this is typically called a butterfly structure, as shown in Fig 5.

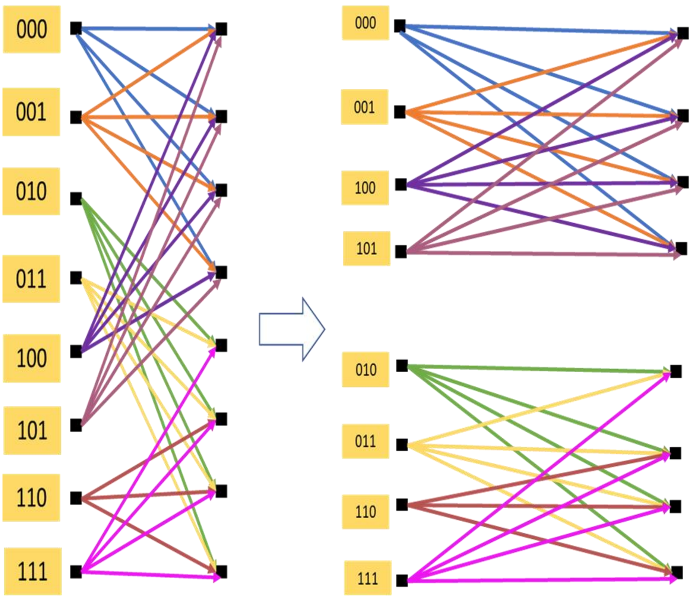

Fig 5: Butterfly structure of (3,2,2) trellis.

Information Sequence Output Decisions

This block is used to release decoded symbols. If a truncation length T is assumed, then at a given time t the decoder needs to release an n-bit symbol decoded at the time (t-T). A decoded sequence error that occurs as a result of the release of the decoded symbols before the entire received sequence has been decode is called a truncation error (T>=5m)[6].

4. SURVIVOR PATH MEMORY DESIGN

There are two basic design approaches : Register Exchange and Traceback.

4.1 Register Exchange Path Memory

The register exchange technique stores forward labels in its path memory, it is known as a forward labels decoder. Fig 6 represent (3,2,2) RE path memory update[9].

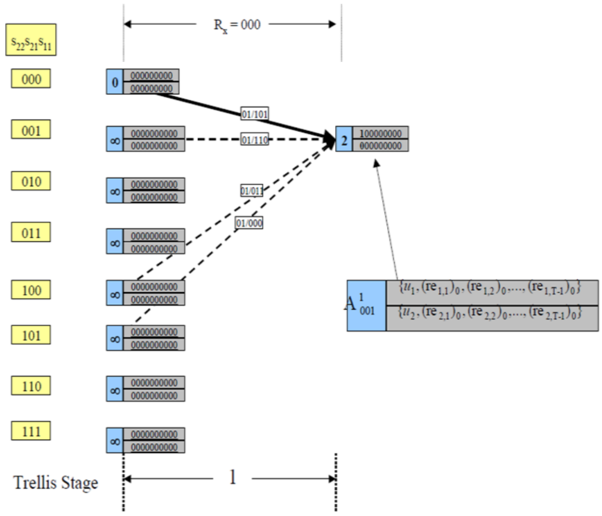

Fig 6: (3,2,2) RE path memory update.

4.2 Traceback Path Memory

Once the path memory is full, decoded symbols are produced by recalling the trellis connections in reverse order. This decoding method is called the traceback method[8].

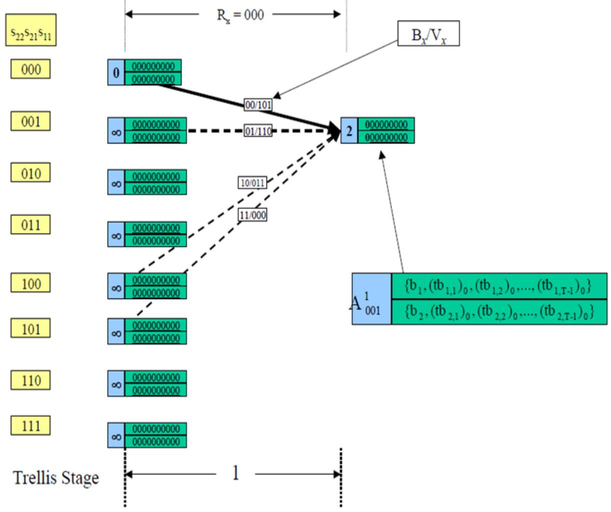

Fig 7: (3,2,2) Traceback Path memory update

4.3 Proposed Path Memory

Proposed Method can achieve Significant savings in storage Memory of Viterbi decoders. This can be achieved by introducing a small modification in the backward label traceback algorithm.

During traceback mapping, the traceback shift register operations work right to left by shifting backward labels from path memory, and producing a predecessor state on every clock cycle back along the surviving path. However

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

when using the Proposed method , the same elements are available mj stages earlier, and the decoded symbol elements are typically resident in the right-most elements of the traceback register as shown in the Fig 8.

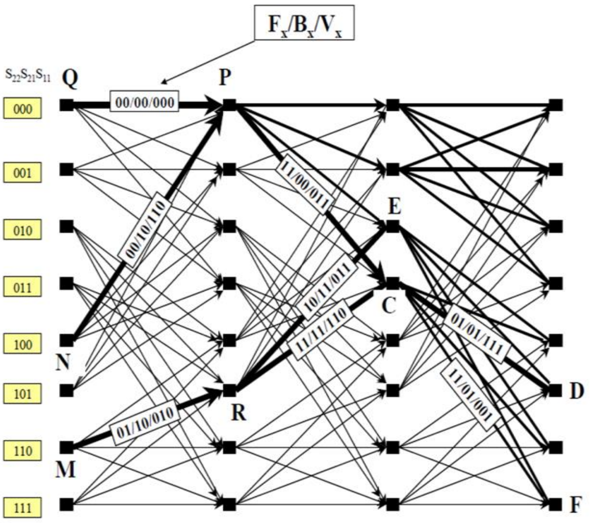

Fig 8: Novel Path Memory trellis structure

5. IMPLEMENTATION

The block diagram for both hardware based (3,2,2) traceback decoders and the associated top-level symbol is shown in Fig 9, followed by the signal descriptions in Table 2.

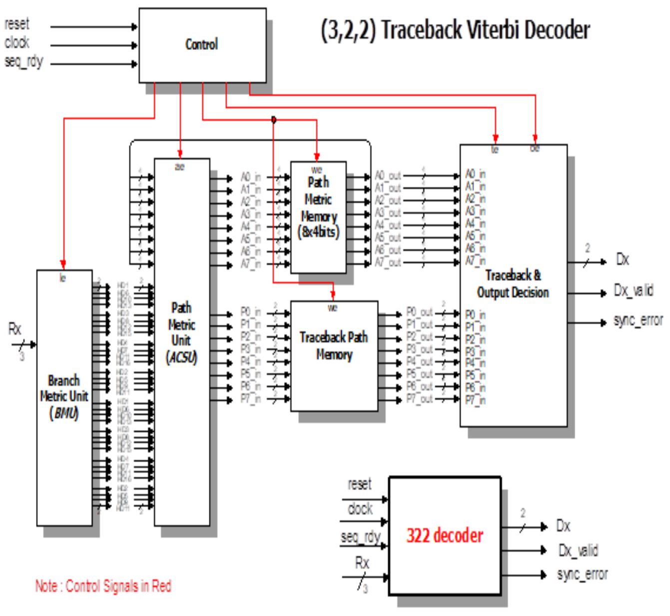

Fig 9: Hardware Viterbi Decoder

Signal	Direction	Description
Rx[2:0]	Input	A 3-bit digital sequence representing the n-bit received sequence for the current trellis stage.
seq_rdy	Input	A pulse to indicate that a new received sequence is ready to be decoded.
clock	Input	Chip level clock input.
reset	Input	Chip level reset input.
Dx[1:0]	Output	The 2-bit decoded symbol output. A valid decoded symbol is qualified by the assertion of the Dx_Valid signal. Otherwise the signal remains in the tri-state condition.
Dx_Valid	Output	An output signal used to qualify a valid decoded symbol is available.
seq_error	Output	An output signal to indicate that the decoder is out of sync with the received sequence.

Table 2: Signal Description for the (3,2,2) Decoder

Branch Metric Unit(3,2,2)

For the (3,2,2) case, the building block is instantiated eight times to compute each of the eight possible branch labels {000-111} for the entire BMU using control enables signal (*le*) as shown in Fig 10.

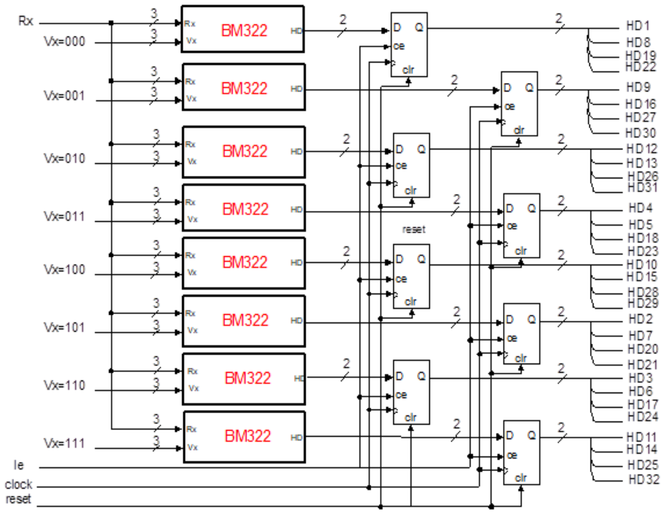

Fig 10: BMU unit

Add Compare Select Unit (ACSU)

If the add enable signal (*ae*) is asserted each incoming path metric and the corresponding branch metric are added together. All sums from the Add operation are then fed into a ≤ comparator[7], with the output of the comparator generating the surviving backward label output. ACS building block is shown in Fig 11.

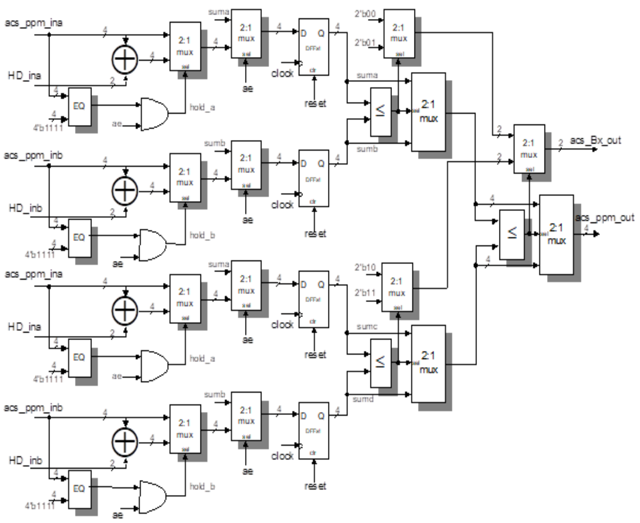

Fig 11: ACS Unit

Conventional vs Proposed Path Memory Storage Unit and Traceback output Decision unit

The (3,2,2) conventional path memory is organized into sixteen parallel shift registers (P0[1],P0[0]-P7[1],P7[0]), each of length ten[11].

The organization of the Conventional path memory shift registers is shown in Fig 12.

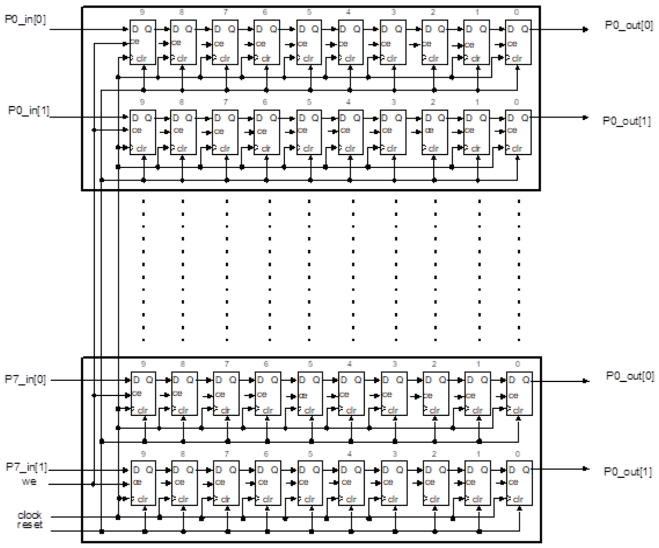

Fig 12: Conventional Path Memory

Once the traceback pointer terminates, the control block asserts the output enable (*oe*) signal to the tri-state output to release a decoded symbol from the *s21* and *s11* of the traceback register elements[10]s. The block diagram of the output decision block is shown below in Fig 13.

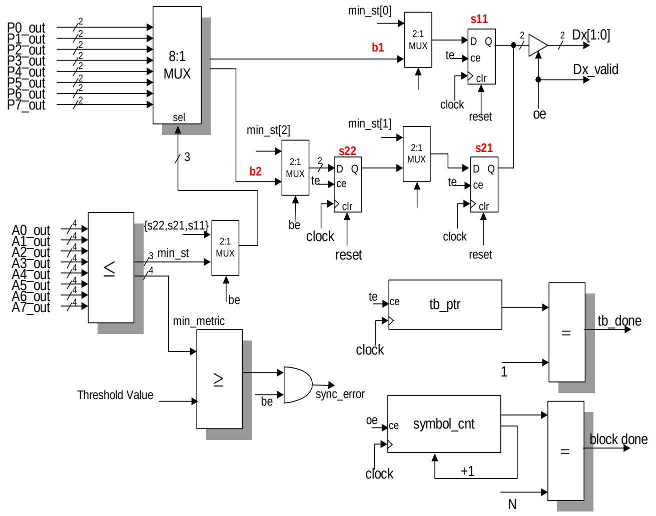

Fig 13: Conventional Traceback Output Decision Unit

The Proposed path memory is organized into sixteen parallel shift registers (P0[1],P0[0]-P7[1],P7[0]), each of length nine and eight, Fig 14 described the Proposed Path Memory unit to achieve storage efficiency.

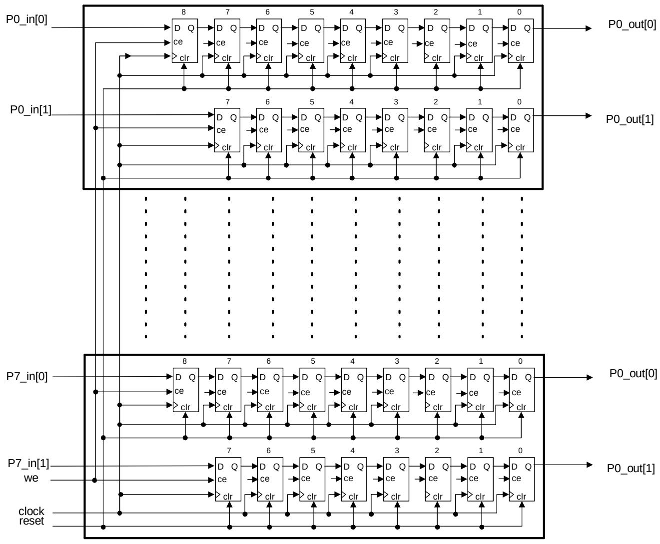

Fig 14:Prposed Path Memory.

The proposed traceback operations terminate earlier than the conventional case because of the smaller memory requirements. In addition, the output decision circuit is wired differently than the conventional case in order to take advantage of the efficient traceback architecture. The Proposed output decision circuit is shown in Fig 15.

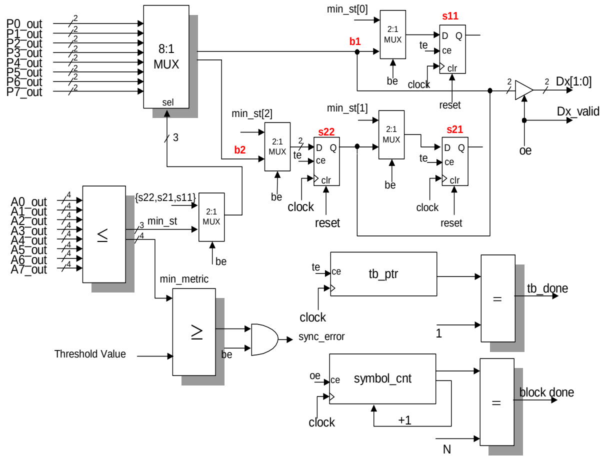

Fig 15: Efficient Traceback Output Decision Unit

Hardware Design - Control Unit

The control unit for the Viterbi decoder is designed to control the entire data path from calculating branch metrics in the BMU to the traceback process to releasing decoded symbols. The control unit moves data in a pipelined fashion through the decoder by enabling and disabling control signals in sequence to the appropriate blocks[10]. The hardware design for the sequencer is carried out by the use a finite state machine FSM.

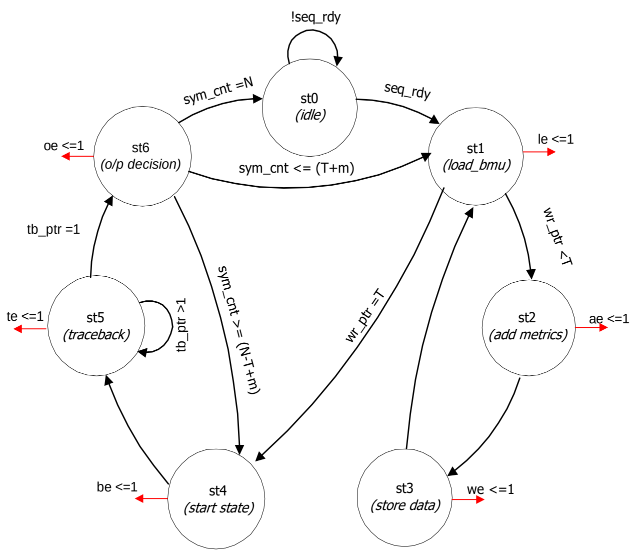

Fig 16: Viterbi Decoder State Diagram

FLOW CHART

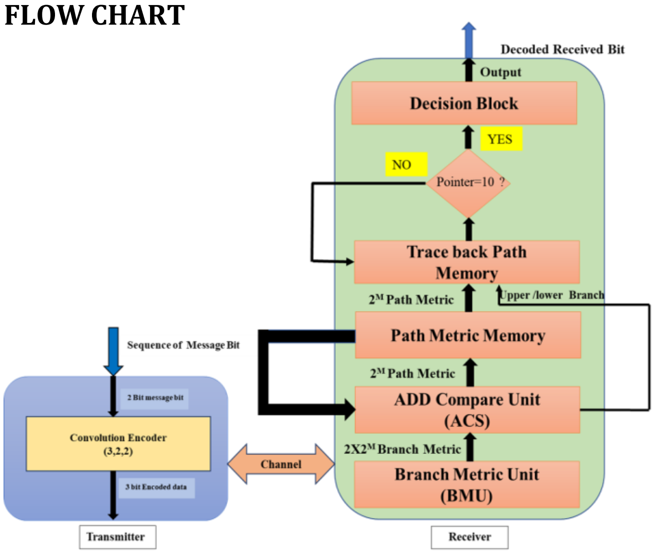

Fig 17: Detail Flow Chart of Viterbi Decoder

6. RESULTS

The design of Viterbi Decoder is carried out using Verilog HDL, Verilog 2001 standard is used for developing a source code and verification is carried out in both ASIC flow And FPGA flow.

6.1 ASIC DESIGN METHODOLOGY RESULTS

Both Conventional and Proposed Viterbi Decoder are simulated using **NCSim** tools of **Incisive Enterprise Simulator** as shown in the following Fig 19 and Fig 20 with minimum output is 1600 ns for conventional method and 1400ns for Proposed method.

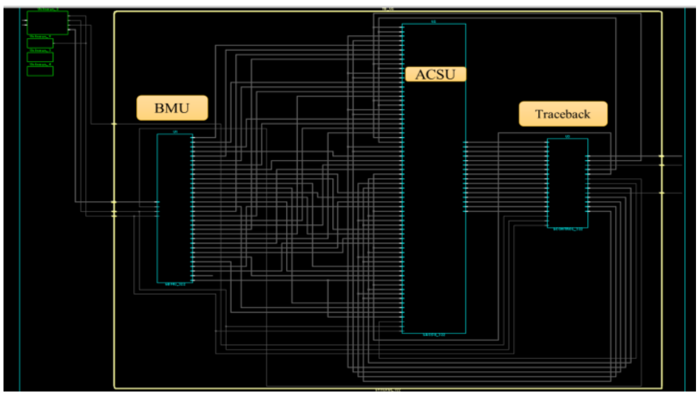

Fig 18:Schematic of Viterbi Decoder

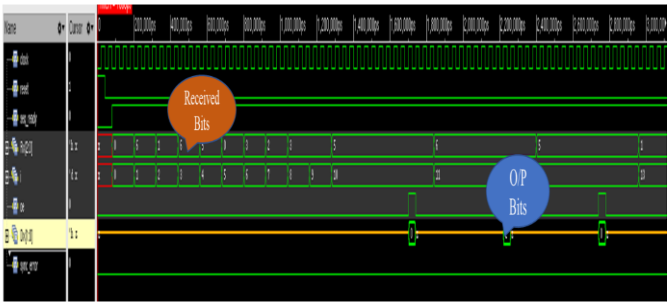

Fig 19:Simulation waveform of Conventional Viterbi Decoder

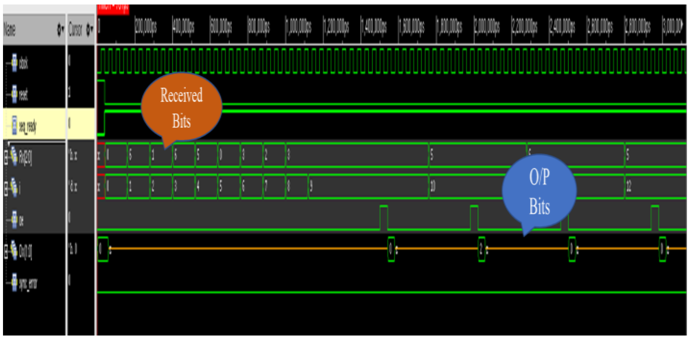

Fig 20: Simulation waveform of Proposed Viterbi Decoder

Difference Between Conventional and Proposed Viterbi Decoder Using Simcompare

Comparison Between Conventional and Proposed Viterbi Decoder Simulation was Carried out with the help of Simcompare Manager as shown in Fig 21.

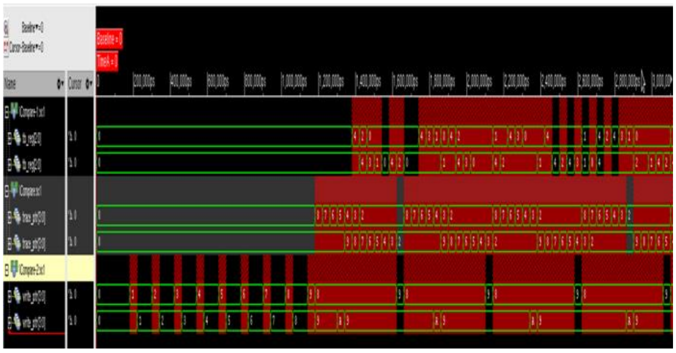

Fig 21: Simulator Comparison Using Simcompare.

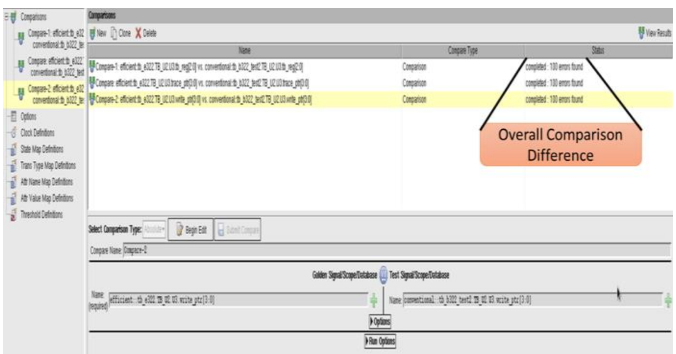

Fig 22: Result in Simcompare Manger.

Code Coverage

Code Coverage is the percentage of code which is covered by the automated tests which is carried out using IMC (Incisive Metrics Center) and result for better coverage are as follows for both Convolution Encoder and Proposed Viterbi Decoder.

Table 3: Code Coverage for Viterbi Decoder.

Type of Coverage	Percentage of Coverage	Improved Coverage
Code	83.9	91.24
Block	91.49	95.62
Toggle	75.73	85.68
FSM	95	100
Overall	84.16	91.46

Logical Synthesis Using Genus

Logic Synthesis of the design Viterbi Decoder was performed using Genus Synthesis Solutions. The design was synthesized with a clock frequency of 100 MHz,65nm Technology library was utilized to synthesize the design . Summary of synthesis reports, that is timing, area and power reports has been tabulated in the Table 4.

Table 4: Synthesis Reports for Viterbi Decoder.

Synthesis Report	Conventional Viterbi Decoder	Proposed Viterbi Decoder
Timing	9.76 ns	9.76 ns
Power	64.87μW	64μW
Area	698μm^2	697μm^2
Number of Cell Used	198	194

6.2. FPGA DESIGN METHEDOLOGY RESULTS

Questa Simulator has been used as to simulate the design for FPGA Flow. Fig 24 an Fig 25 represents simulation waveform of design Conventional and a proposed Viterbi Decoder respectively.

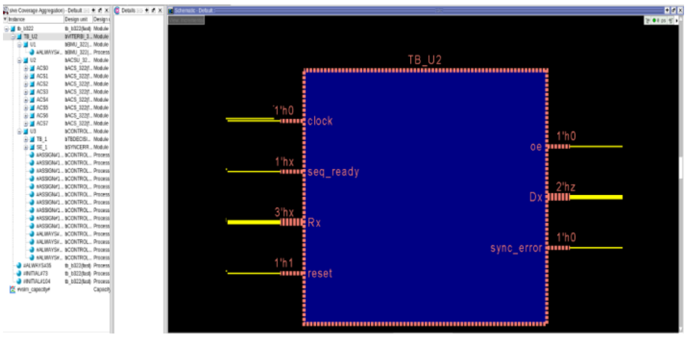

Fig 23:RTL Schematic of Viterbi Decoder.

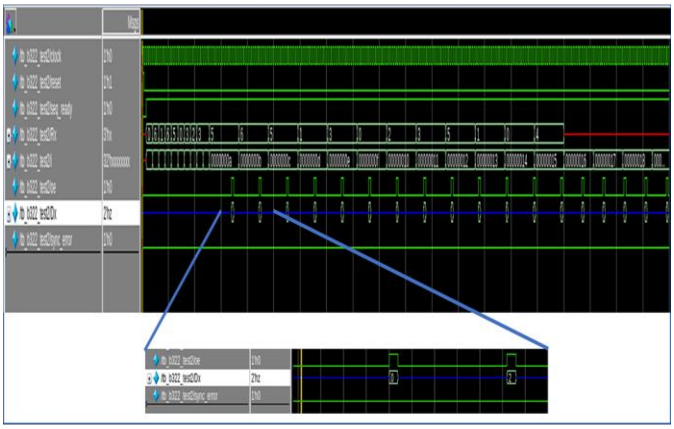

Fig 24: Simulation Waveform OF Conventional Viterbi Decoder

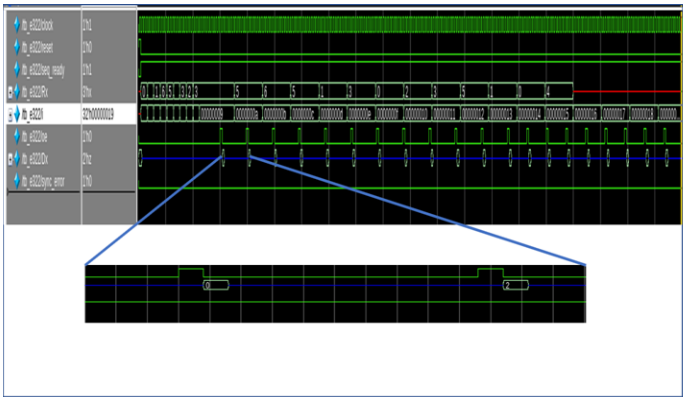

Fig 25: Simulation Waveform of Proposed Decoder

Synthesis Using Precision RTL

The FPGA device ZYNQ 7Z020CLG400 belonging to Xilinx vendor, with speed grade -1 was selected for the Synthesis using Precision RTL.

Table 5: Synthesis Reports for Viterbi Decoder

Synthesis Report	Parameter	Conventional Decoder	Proposed Decoder
Timing	Cell Delay	30.02%	29%
	Net Delay	69.98%	70%
	Slack	3.798	3.733
Area	Total number of DFFs	262	260
	Total number of LUTs	568	492
	Total number of accumulated Instances	865	787

Table 6: Utilization Comparison for Viterbi Decoder parameter

Module	Conventional Design		Proposed Design	
	LUTs	DFFs	LUTs	DFFs
BMU	14	12	14	12
ACSU	348	108	348	108
Path Memory	0	20	0	16
Control, Traceback, & Output Decision Logic	206	122	130	124
TOTAL	568	262	492	260

Equivalence Check Results Using Formalpro

FormalPro tool has been used to check the functional equivalence between RTL and gate-Level Netlist. As shown in the Fig 26.

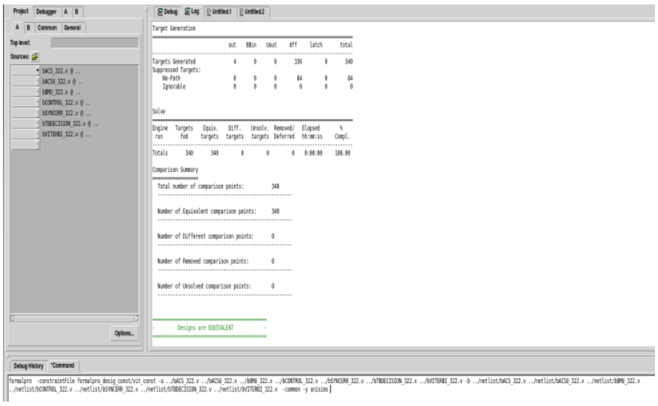

Fig 26: Formal Equivalence Check

7. CONCLUSIONS

The different blocks of Viterbi Decoder were successfully simulated and synthesized with continuous optimization. An above 90% of code coverage and optimized power, Area was Compared between Conventional and Proposed method with the minimum timing possible, reducing the overall working frequency to 100 MHz . The design works well within the timing constraint of 10ns (9.67ns).

8. FUTURE WORK

This design can be further optimized to obtain better coverage and optimized power and timing. It can be taken to further steps in the ASIC design flow namely, Physical Design.

REFERENCES

[1] B. Sklar, Digital Communications: Fundamentals and Applications. Prentice-Hall Inc., Englewood Cliffs, N.J., 1988

[2] G.D. Forney Jr. The Viterbi Algorithm. Proc. IEEE Vol. 61, 1973, pp. 268-278.

[3] Ms.G.S. Suganya and Ms. G.Kavya "RTL Design and VLSI Implementation of an efficient Convolutional Encoder and Adaptive Viterbi Decoder", International conference on Communication and Signal Processing , 2013.

[4] Amruta J. Mandwale and Altaf O. Mulani "Different Approaches For Implementation of Viterbi decoder on reconfigurable platform", International Conference on Pervasive Computing, 2015.

[5] Khloud Mostafa, Aziza Hussein, Hassan Youness , Mohammed Moness "High Performance Reconfigurable Viterbi Decoder Design for Multi-Standard Receiver", National Radio Science Conference, 2016.

[6] J.Nargis, D.Vaithiyanathan, R.Seshasayanan "Design of High Speed Low Power Viterbi Decoder for TCM System", 2016.

[7] I. Swathi and S. Rajaram "FPGA Implementation of Viterbi Decoder for Software Defined Radio Applications", 2017.

[8] M. Ruban Gladwin and N. Kasthuri—"An Ultra Low Power VLSI Architecture for Viterbi Decoder using Subthreshold Adiabatic Logic", 2018 IEEE.

[9] Raj Mamarde and Suchitra Khoje "Viterbi Decoder using Zynq-7000 AP-SoC", 2018 IEEE.

[10] Xuying Zhao, Huan Li and Xiaoqin Wang "A High Performance Multi-standard Viterbi Decoder", 2017 IEEE.

[11] Surekha K. Tadse and Dr. S. L. Haridas " A Low Power Asynchronous Viterbi Decoder", International Conference on Emerging Trends in Engineering and Technology,2019.