http://www.jctjournals.com ISSN(ONLINE):2278 - 3814

Volume-12, Issue-06, June2023
JOURNAL OF COMPUTING TECHNOLOGIES (JCT)
International Journal
Page Number: 01-06

DESIGN AND IMPLEMENTATION OF AN ENHANCED - USING VITERBI DECODER

Gaurav Patil [1] Devendra Patle [2],
[1]M. Tech student VLSI, [2]Assistant Professor,
[1,2]Department of Electronics and Communication
[1,2] School of Engineering, SSSUTMS Sehore (M.P), INDIA,

***Abstract*—In this paper discuss about design and implementation of an enhanced viterbi decoder using VHDL with improved performance for SDN. The increase in demand of transaction in different devices require secure communication channel between transmitter and receiver. For the secure communication channel apply channel coding between sender and receiver end. In this research work first discuss on different viterbi decoder that is presented by different researchers in the last decade, describe in the literature survey. Also discuss the viterbi decoder and parts in technical background. The proposed viterbi doeoder is design by three different parts Branch Metric Unit, Path Metric Unit, Survivor Memory unit shown in the decoder. For the simulation of proposed viterbi decoder use Xilinx 14.1 and for synthesis of proposed design use I-sim. The proposed design successful synthesis and simulated without any error and proposed design shows low area in terms of number of slices , number of flip flops, and number of LUTs, also calculate the delay that is very low.**

Keywords*— Branch Metric Unit, Path Metric Unit, Survivor Memory unit, look up tables (LUTs), number of slices, number of Gates and delay, viterbi decoder, Xilinx 14.1, I-Sim, forward error coding , Block coding and convolutional coding.

I. INTRODUCTION

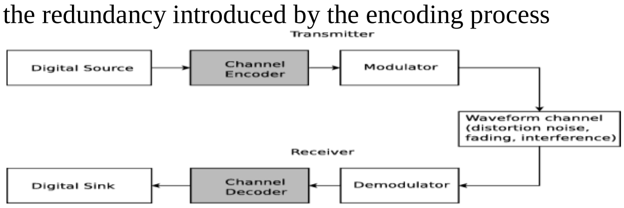
The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without re-transmission. FEC gives the receiver the ability to correct errors without needing a reverse channel to request re-transmission of data, but at the cost of a fixed, higher forward channel bandwidth. FEC is therefore applied in situations where re-transmissions are costly or impossible, such as one-way communication links and when transmitting to multiple receivers in multicast. For example, in the case of a satellite orbiting around Uranus, a re-transmission because of decoding errors can create a delay of 5 hours FEC information is usually added to mass storage (magnetic, optical and solid state/flash based) devices to enable recovery of corrupted data, is widely used in modems, is used on systems where the primary memory is ECC memory and in broadcast situations, where the receiver does not have capabilities to request retransmission or doing so would induce significant latency.

FEC processing in a receiver may be applied to a digital bit stream or in the demodulation of a digitally modulated carrier. For the latter, FEC is an integral part of the initial analog-to-digital conversion in the receiver. The Viterbi decoder implements a soft-decision algorithm to demodulate digital data from an analog signal corrupted by noise. Many FEC coders can also generate a bit-error rate (BER) signal which can be used as feedback to fine-tune the analog receiving electronics.

1.1 Forward Error Correction (Channel Coding)

Forward Error Correction (channel coding) is a powerful technique for increasing the transmission system margin. For example, with the standardized Reed-Solomon code used in submarine systems [8], a BER of lower than 10^{-13} can be obtained for a BER before correction of only 10^{-4}, thus providing a 5.8-dB system margin.

The channel encoder introduces, in a controlled manner, some redundancy in the binary information sequence that can be used at the receive side to check and correct errors. More precisely, the channel encoder transforms a sequence of *k* information symbols into a unique n-symbol sequence, called a code word. The ratio *k/n* is called the code rate. The inverse of the code rate, namely *n/k*, is a measure of the redundancy introduced by the encoding process



ISSN(ONLINE):2278 – 3814

Fig 1 basic model of a digital transmission system using FEC techniques.

II. PROPOSED METHOD

The Viterbi algorithm has been known as a maximum likelihood decoding algorithm for convolutional codes. Let us consider a simple example for illustrating the principle of Viterbi algorithm. Assume that a car that has 3 states forward, stop and reverse with the condition that a transition from forward to reverse is not allowed. In other words, it implies that the car first enter the stop state and then enter the reverse state. Hence, when we receive the information through the processes of forward, reverse and stop, we can safely interpret it as – forward, stop and reverse as this is a "maximum likelihood sequence". The Viterbi algorithm uses the trellis diagram to compute the path metric value (accumulated distance) from the received sequence to the possible transmitted sequences. The total number of such trellis paths increases exponentially with the number of stages in the trellis. It causes potential complexity and memory problems. The Viterbi decoding algorithm has been classified into hard decision decoding and soft decision decoding. If the received signal is converted into two levels, either zero or one, it is called hard decision. If the input signal is quantized and processed for more than two levels, it is called soft decision. The soft decision decoding is expensive and require large amount of memory than hard decision decoding.

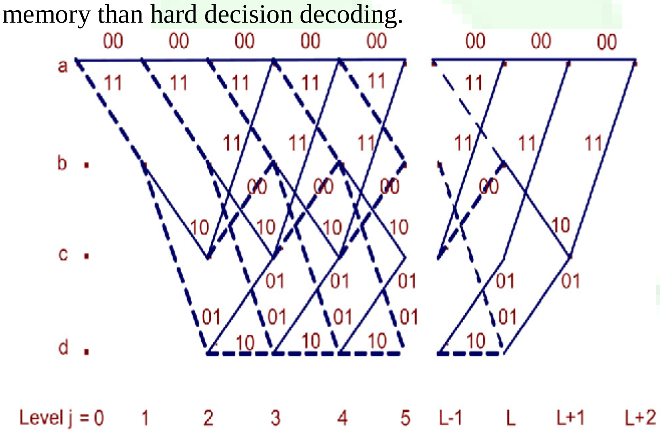


Fig 2 Hard decision Viterbi decoding trellis diagram

Viterbi Decoder

Viterbi algorithm is used in the Viterbi decoder for decoding a bit stream that has been encoded using FEC based on a Convolutional code. Figure 5 shows the block diagram of Viterbi decoder. It consists of the following functional units, namely, Branch Metric Unit, Path Metric Unit, Survivor Memory unit.

A. Branch Metric Unit

branch metric unit's function is to calculate *branch metrics*, which are normed distances between every possible symbol in the code alphabet, and the received symbol. There are hard decision and soft decision Viterbi decoders. A hard decision Viterbi decoder receives a simple bit stream on its input, and a Hamming distance is used as a metric. A soft decision Viterbi decoder receives a bit stream containing information about the *reliability* of each received symbol. For instance, in a 3-bit encoding, this reliability information can be encoded as follows:

Table 1 Shows a 3-bit encoding

value	meaning	
000	strongest	**0**
001	relatively strong	**0**
010	relatively weak	**0**
011	weakest	**0**
100	weakest	**1**
101	relatively weak	**1**
110	relatively strong	**1**
111	strongest	**1**

The comparison between received code symbol and expected code symbol is done by branch metric unit. It also counts the number of differing bits. It is the smallest unit in the Viterbi decoder. The measured value of the BMU can be the Hamming distance in case of the hard input decoding or the Euclidean distance in case of the soft input decoding.

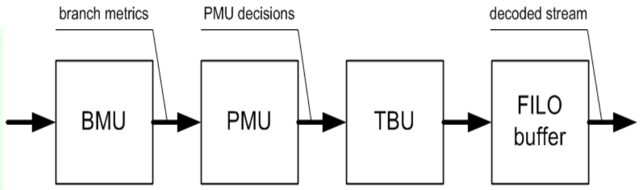


Fig 3 a. A common way to implement a hardware viterbi decoder

Branch metric unit (BMU)

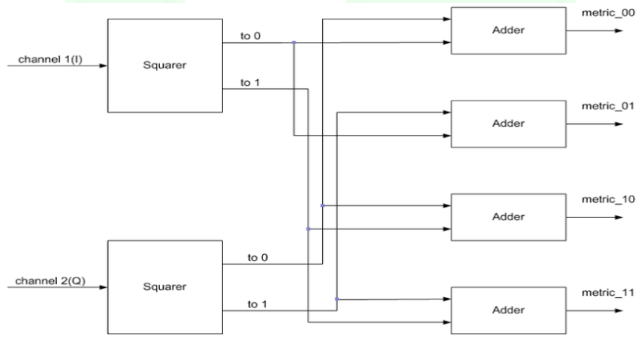


Fig 3 b. A sample implementation of a branch metric unit

III. SOFTWARE IMPLEMENTATION

For the implementation of proposed viterbi decoder using Xilinx ISE design suit software version 14.1. In Xilinx software provide two platform for the implementation of hardware descriptive language Verlog and VHDL. In this proposed work for the implementation of viterbi decoder use VHDL platform for the implementation and synthesis of proposed viterbi decoder use i-sim simulator. VHDL is mainly used to point the function of a circuit. Text models using VHDL which depicts a logic circuit that is refined by a synthesis program. The logic design is tested by using a simulation program. The design is interfaced by the logic circuits using the simulation models. The proposed design in VHDL IDE to produce the RTL schematic of the desired circuit. After that, the generated schematic may be verified exploitation simulation software system that shows the waveforms of inputs and outputs of the circuit. In the below section shows some important of VHDL implementation.

A. Design Entry

This is the first step of implementing a design on field programmable gate array. In this step the VHDL (Very High Speed Integrated Chip Description Language) code of viterbi algorithm implementation Architecture was written using software Xilinx ISE 14.1. Structural modeling was used for writing the code. After writing the code syntax check was performed on the code to see whether code was properly written using correct syntax.

B. Behavioral Simulation

The next step is behavioral simulation. This step verifies whether the design entered is functionally correct or not. This simulation is called RTL simulation. For this simulation VHDL Test bench was written for image algorithm implementation architecture and simulation was seen in Xilinx ISE Simulator. After it is verified it is functionally correct we move onto next step.

C. Design Synthesis

The VHDL code of image algorithm implementation is then synthesized using Xilinx XST which is a part of Xilinx ISE software. There is an option of Synthesis in process tab of Xilinx ISE which performs the operation of synthesis .The synthesis process is used for optimizing the design architecture selected. The resulting net list is saved to an NGC file. After design synthesis, synthesis report is generated which gives information about how many logic blocks are used and what is the device utilization of the design architecture synthesized. Synthesis basically maps the behavioral design to gate level design.

D. **Design Implementation**

After design synthesis design implementation is done which comprises of following three steps

(a) Translate

(b) Map

(c) Place and Route

Before translating the design, User Constrained file (UCF) is written to assign pin configuration of the field programmable gate array to the viterbi implementation I/O‟s. Once this is done Translate merges together this UCF file and net list generated after synthesis into Xilinx design file Mapping is done to fit the design into the available resources of target device i.e. field programmable gate array. This is also important step of design. Last step of Design Implementation is Placing and Routing which places the logic blocks of the design into field programmable gate array and route them together so that they occupy minimum area and meet timing requirements. This operation produces NCD output file. In the below figure 5.4 shows project summery report that is shows that is successfully implemented there is no error occurs in synthesis report. This summery report also shows used number of Slices, number of Slice Flip Flops, number of 4 input LUTs, number of bonded IOBs and number of GCLKs utilized in this project.

```
=========================================================================
*                          Final Report                               *
=========================================================================
Final Results
 RTL Top Level Output File Name                              : viterbi.ngr
Primitive and Black Box Usage:
------------------------------
# BELS                      : 81
#      GND                  : 1
#      INV                  : 1
#      LUT2                 : 14
#      LUT3                 : 1
#      LUT4                 : 12
#      LUT5                 : 19
#      LUT6                 : 28
#      MUXF7                : 5
# FlipFlops/Latches         : 44
#      FD                   : 44
# Clock Buffers             : 1
#      BUFGP                : 1
# IO Buffers                : 140
#      IBUF                 : 3
#      OBUF        137
:========================================================================
=
Device utilization summary:
Selected Device : 3s100evq100-5
Slice Logic Utilization:
 Number of Slice Registers:            32  out of  11440     0%
 Number of Slice LUTs:                 75  out of   5720     1%
    Number used as Logic:              75  out of   5720     1%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     79
   Number with an unused Flip Flop:      47  out of     79    59%
   Number with an unused LUT:             4  out of     79     5%
   Number of fully used LUT-FF pairs:    28  out of     79    35%
   Number of unique control sets:         1
=========================================================================
```

Total Delay

```
Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 5455 / 33
    ---------------------------------------------------------------------
Offset:            9.145ns (Levels of Logic = 8)
Source:            data_in<0> (PAD)
Destination:       global_distance_3_1 (FF)
 Destination Clock: clk rising
 Data Path: data_in<0> to global_distance_3_1
                         Gate     Net
   Cell:in->out    fanout  Delay   Delay  Logical Name (Net Name)
     ----------------------------------------  ------------
     IBUF:I->O          21   1.222   1.218  data_in_0_IBUF (data_in_0_IBUF)
     LUT2:I0->O          9   0.203   1.194  Mram__n03261 (Madd_n0181_lut<0>)
     LUT6:I0->O          1   0.203   0.684  Mmux__n0228393 (Mmux__n0228393)
     LUT6:I4->O          6   0.203   0.745  Mmux__n0228394 (_n0236<2>)
     LUT5:I4->O              14      0.205     0.958   BUS_0042_BUS_0045_LessThan_47_o3
   (BUS_0042_BUS_0045_LessThan_47_o)
     LUT6:I5->O                         5        0.205        0.943     Mmux__n02283111
   (branch_distance[0][2]_branch_distance[0][2]_mux_61_OUT<0>)
     LUT6:I3->O                                     3            0.205            0.651
   Msub_GND_5_o_GND_5_o_sub_73_OUT_lut<0>1(Msub_GND_5_o_GND_5_o_sub_73_
   OUT_lut<0>)
     LUT6:I5->O          1   0.205   0.000  Mmux_n017121 (n0171<1>)
     FD:D               0.102          global_distance_2_1
   ----------------------------------------------------------------------------
     Total                9.145ns (2.753ns logic, 6.392ns route)
                              (30.1% logic, 69.9% route)
   ============================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 21 / 21
-------------------------------------------------------------------------

Offset:          3.701ns (Levels of Logic = 1)
Source:          survivors_0 (FF)
Destination:     survivors0<0> (PAD)
Source Clock:    clk rising
                       Gate      Net
   Cell:in->out    fanout   Delay   Delay  Logical Name (Net Name)
     FD:C->Q            4   0.447   0.683  survivors_0 (survivors_0)
   OBUF:I->O              2.571          survivors0_0_OBUF (survivors0<0>)-------------------
                                                          --------------
```

Total = 3.701ns (3.018ns logic, 0.683ns route)(81.6% logic, 18.4% route)

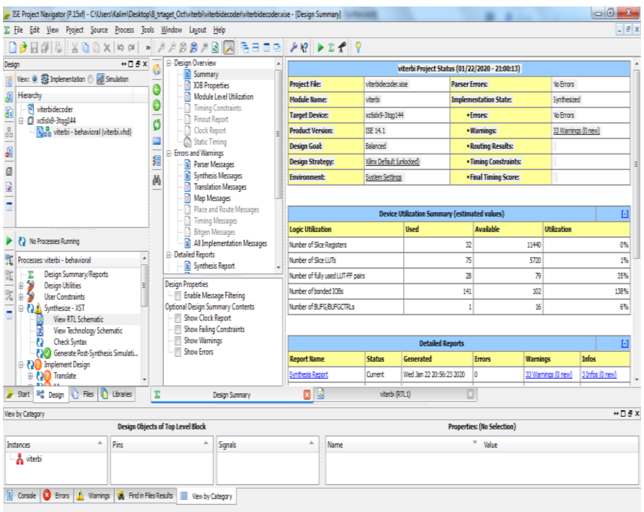


Fig. 4 Viterbi Decoder project summery Report

ISSN(ONLINE):2278 - 3814

In the above section discuss the design summery of proposed viterbi decoder now discuss about the RTL schematics of viterbi decoder, in the RTL schematics shows input and output of proposed design IC. In the proposed design as a input used data. clock and reset. The input data is encoded data that is decoded by viterbi decoder and shows the decoded data in data_out.

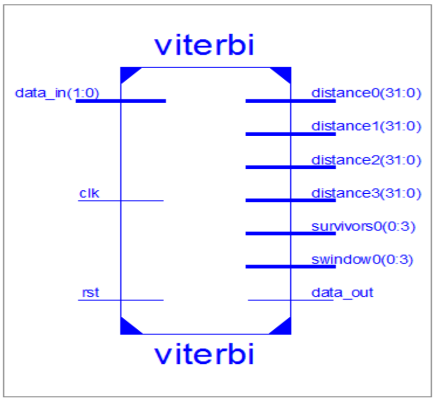


Fig. 5 RTL Schematics

In the above section discuss about the logic level implementation of viterbi decoder also discuss the number of used components of design, and discuss the logic delay. In the next chapter discuss about the Simulation of viterbi decoder.

IV. SIMULATION RESULT OF PROPOSED WORK

The Viterbi decoder takes the distance measures and calculates the most likely transmitted signal. It does this by keeping a running history of the previously received signals in a path memory. The path-memory length of this decoder is 12. By keeping a history of possible sequences and using the knowledge that the signals were generated by a state machine, it is possible to select the most likely sequences.

I-SIM Simulator Result Wave Form

The system input or message, Data_in[1:0] , is driven by a counter that repeats the sequence 0, 3,2,0,1,0,1,3,1,0 ... by random number at each positive clock edge (with a delay of one time unit), starting with X equal to 4 at t= 0. The active-high reset signal, Res, is asserted at t = 40. The encoder output, uncoded_signal[2:0], changes at t = 60, which is one time unit (the positive-edge-triggered D flip-flop model contains a one-time-unit delay) after the first positive clock edge (at t = 99) following the dissertation of the reset at t = 1000ns. The encoder output sequence beginning at t= 62 is and then the sequence 0, 3,2,0,1,0,1,3,1,0 ... . This encoder output sequence is then imagined to be transmitted and received.

The Viterbi decoder model presented in this work is written for both simulation and synthesis. The Viterbi decoder makes extensive use of vector D flip-flops (registers). Early versions of VHDL did not support vector instantiations of modules. In addition the inputs of UDPs may not be vectors and there are no primitive D flip-flops in VHDL. This makes instantiation of a register difficult other than by writing a separate module instance for each flip-flop

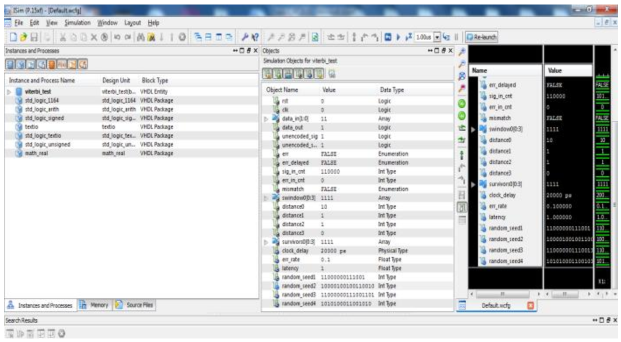


Fig. 6 Shows the I-Sim Simulator output

Instance and Process Name	Design Unit	Block Type
viterbi_test	viterbi_test(b...	VHDL Entity
std_logic_1164	std_logic_1164	VHDL Package
std_logic_arith	std_logic_arith	VHDL Package
std_logic_signed	std_logic_sig...	VHDL Package
textio	textio	VHDL Package
std_logic_textio	std_logic_tex...	VHDL Package
std_logic_unsigned	std_logic_un...	VHDL Package
math_real	math_real	VHDL Package

Fig. 7 Shows the Different Lib. used in decoder

There are different VHDL lib. used for simulation of the proposed decoder math_real.vhd, std_logic_1164.vhd, std_logic_arith.vhd,std_logic_signed.vhd,std_logic_textio. vhd, std_logic_ unsigned.vhd, textio.vhd and viterbi.vhd.

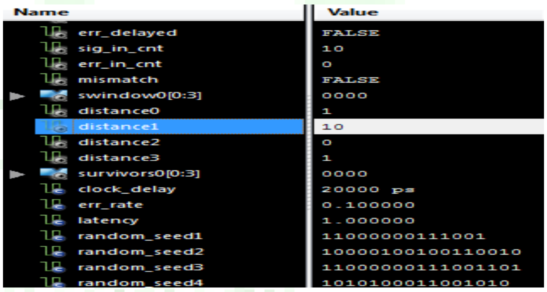


Fig. 8 Shows the simulate decoder

These are the source files use to simulate decoder. Reset rst, clock clk, data_in, data_out, unencoded_sig, unencoded_sig_delayed, err, sig_in_cnt, mismatch these are the objects for veterbi decoder. Both simulation objects files and VHDL source lib. files are shown in the above figure 8For the signal generation used random signal which is output shown in above figure 8 wave form random seed 1, random seed 2, random seed 3 and random seed 4

\

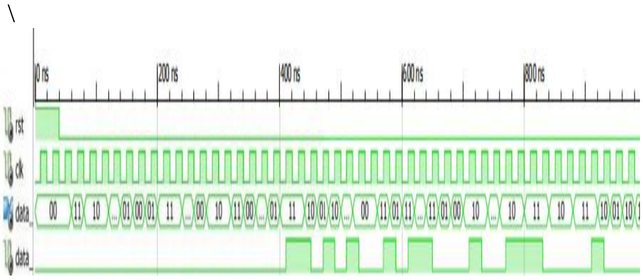


Fig. 9 Shows the Input data in Binary form (Wave Form)

In the above figure 9 data_in[1:0] shows the input signal and data encoder wave form to the proposed encoder.

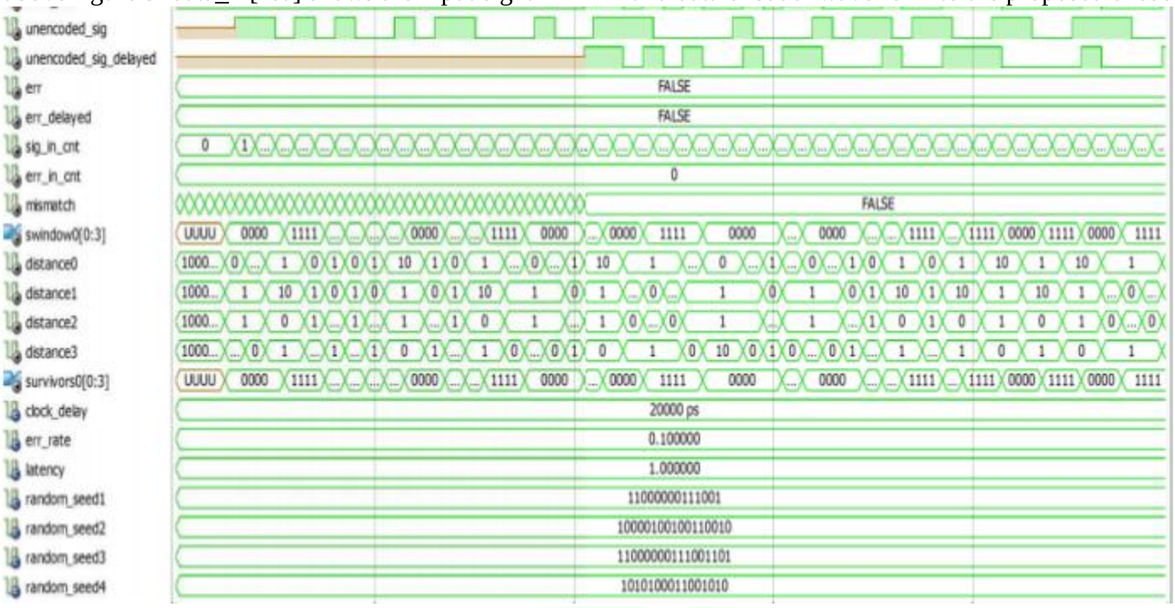


Fig. 10 shows the resultant decoder output

In the above figure 10 shows the decoder output resultant.

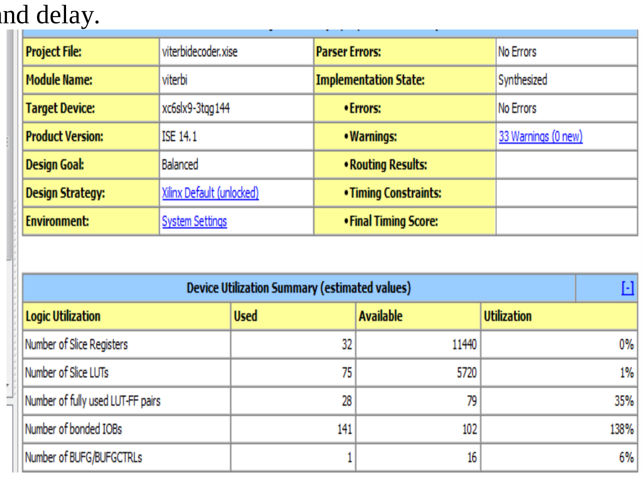
V. RESULT COMPARISON

In the above section discuss the simulation outcomes on i-sim simulator now discuss the result comparison of proposed decoder compare with previous decoder. That is shown in below Table II.

Table II Shows Comparison of Proposed Method With Different Previous Works

Year	Paper	Area in number of LUTs			
		LTUs	Slices	Flip Flop	GCLCK
2023	Proposed - Vetribi Decoder	75	32	28	1
2022	FPGA Implementation of Viterbi Decoder for Software Defined Radio Applications	118	65	43	1

In the below figure shows the comparison of proposed work with base paper on different result parameters such as look up tables (LUTs), number of slices, number of Gates and delay.



Project File:	viterbidecoder.xise	Parser Errors:	No Errors
Module Name:	viterbi	Implementation State:	Synthesized
Target Device:	xc6slx9-3tqg144	•Errors:	No Errors
Product Version:	ISE 14.1	•Warnings:	33 Warnings (0 new)
Design Goal:	Balanced	•Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	•Timing Constraints:	
Environment:	System Settings	•Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	32	11440	0%
Number of Slice LUTs	75	5720	1%
Number of fully used LUT-FF pairs	28	79	35%
Number of bonded IOBs	141	102	138%
Number of BUFG/BUFGCTRLs	1	16	6%

Fig. 11 Shows the Proposed Viterbi Device Utilization Summery

TABLE II
COMPARISON OF HARDWARE UTILIZATION.

S. no	Existing method	Device	No of slice LUTs	Percentage of reduction
1	[2]	Virtex 4	172	31%
2	[4]	Virtex 4	131	10%

Fig. 12 Shows the Base paper Viterbi Device Utilization Summery

In the above figure 11 and 12 shows comparison of device utilization summery of proposed viterbi and previous viterbi proposed by different researchers.

IV. CONCLUSION AND FUTURE WORK

In this paper discussed on different viterbi decoder that is presented by different researchers in the last decade, describe in the literature survey. Also discuss the viterbi decoder and parts in technical background. The proposed viterbi doeoder is design by three different parts Branch Metric Unit, Path Metric Unit, Survivor Memory unit shown in the decoder. For the simulation of proposed viterbi decoder use Xilinx 14.1 and for synthesis of proposed design use I-sim. The proposed design successful synthesis and simulated without any error and proposed design shows low area in terms of number of slices , number of flip flops, and number of LUTs, also calculate the delay that is very low. These three parameters i.e. power, area and speed are always traded off. However, area and speed are usually conflicting constraints, so that improving speed results mostly in larger areas. In the result shows the comparison of proposed work with base paper on different result parameters such as look up tables (LUTs), number of slices, number of Gates and delay and shows better result as compare to previous work. The main aim was to implement convolution encoder and viterbi decoder with code rate 2/3 in compact VHDL.

REFERENCES

[1] Namratha, & Bakhar, M. (2023). Power and area optimized adaptive Viterbi decoder for high speed communication applications. *International Journal of Information Technology*, *15*(1), 45-52.

[2] Devi, T. Kalavathi, E. B. Priyanka, P. Sakthivel, and A. Stephen Sagayaraj. "Low complexity modified viterbi decoder with convolution codes for power efficient wireless communication." *Wireless Personal Communications* 122, no. 1 (2022): 685-700.

[3] Rowshan, M., & Viterbo, E. (2021). List Viterbi decoding of PAC codes. *IEEE Transactions on Vehicular Technology*, *70*(3), 2428-2435.

[4] Sokjabok, Siwakon, Chanon Warisarn, Santi Koonkarnkhai, and Jaejin Lee. "Modified 2D Viterbi algorithm using 2D modulation encoding constraints in BPMR systems." In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp. 457-460. IEEE, 2020..

[5] Harsh, G. Bhuvanendra, and G. Lakshma Reddy. "A VLSI Design of LTE Turbo Encoder-Decoder with Radix 4 ACS Architecture." (2019).

[6] Sharma, Vibhuti, and Sunil Sharma. "Analyzing the Bit Error Rate & Hardware Implementation of Convolution Encoder & Viterbi Decoder."(2019)

[7] Sujatha, E., C. Subhas, and Giri Prasad. "Performance improvement of Turbo Decoder using VLSI Optimization Techniques." 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN). IEEE, 2019.

[8] Gao, Zhen, et al. "Design and Implementation of Configuration Memory SEU-Tolerant Viterbi Decoders in SRAM-Based FPGAs." IEEE Transactions on Nanotechnology 18 (2019): 691-699.

[9] Amarnath, V., et al. "Simulation Improvement of Rising Field Programmable Gate Array Rectify Methodologies by Victimisation Playacting SEC-DAED-TAED-TETRA AED Code." Journal of Computational and Theoretical Nanoscience 16.5-6 (2019): 2362-2367.

[10] Сайлауқызы, Ж., and M. Коккоз. "Research of Noise Immunity of Viterbi Decoder in the Case of Different Depths of Decoding in MATLAB Environment and Projection on FPGA." (2019).

[11] Ahmadinejad, Hosein, Abolfazl Falahati, and Ebrahim Shafiee. "Design and Implementation of a Visible Light Communication and Coding System Based on IEEE802. 15.7 Standard." 2019 2nd West Asian Colloquium on Optical Wireless Communications (WACOWC). IEEE, 2019.

[12] Zbaid, Riham Ali, and Kasim K. Abdalla. "Design and Implementation of Convolutional Encoder and Viterbi Decoder Using FPGA." Journal of University of Babylon 26.3 (2018): 22-29.

[13] Kumari, Dasari Ratna, G. Srinivasa Rao, and K. Anka Siva Prasad. "Low Cost VLSI Architecture for Proposed Adiabatic Offset Encoder and Decoder." (2018).

[14] Thakur, Akash, and Manju K. Chattopadhyay. "Design and Implementation of Viterbi Decoder Using VHDL." IOP Conference Series: Materials Science and Engineering. Vol. 331. No. 1. IOP Publishing, 2018.

[15] Prasad, N., Indrajit Chakrabarti, and Santanu Chattopadhyay. "An energy-efficient network-on-chip-based reconfigurable Viterbi decoder architecture." IEEE Transactions on Circuits and Systems I: Regular Papers 65.10 (2018): 3543-3554.

[16] Adiono, Trio, Ahmad Zaky Ramdani, and Rachmad Vidya Wicaksana Putra. "Reversed-Trellis Tail-Biting Convolutional Code (RT-TBCC) Decoder Architecture Design for LTE." International Journal of Electrical & Computer Engineering (2088-8708) 8.1 (2018).

[17] Tobola, John D., and James E. Stine. "Low-Area Memoryless optimized Soft-Decision Viterbi Decoder with Dedicated Paralell Squaring Architecture." 2018 52nd Asilomar Conference on Signals, Systems, and Computers. IEEE, 2018.

[18] Xiaobo, Jiang, Zhang Fang, and Zeng Zhen. "High-performance Decoder for Convolutional Code with Deep Neural Network." arXiv preprint arXiv:1812.11455 (2018).

[19] Yueksel, Hazar, et al. "Design Techniques for High-Speed Multi-Level Viterbi Detectors and Trellis-Coded-Modulation Decoders." IEEE Transactions on Circuits and Systems I: Regular Papers 65.10 (2018): 3529-3542.

[20] Adiono, Trio, et al. "Design of an OFDM system for VLC with a Viterbi decoder." IEIE Transactions on Smart Porcessing and Computing (SPC) 6.6 (2017): 455-465.

[21] Laddha, Deepali R., Archana O. Vyas, and M. E. Student. "FPGA Implementation of Viterbi Decoder for Long Survivor Path." International Journal of Engineering Science 11822 (2017).

[22] Wankhede, Shweta Anand, and Nilesh Bodne. "Review Paper On Implementation Of Low Power Hard Decision Viterbi Decoder In VLSI." (2017).

[23] Shende, Ms Sneha T., and Asst Prof SK Tadse. "Designing of Asynchronous Viterbi Decoder for Low Power Consumption using Handshaking Protocol: A Technical Review." (2017).

[24] Kadu, Sanket, et al. "Design and Implementation of Viterbi Encoder and Decoder on FPGA." International Journal 3 (2017): 30-33.

[25] Wankhede, Shweta Anand, and Nilesh Bodne. "A Configurable and Low Power Hard-Decision Viterbi Decoder in VLSI Architecture." (2017).

[26] Elwazan, Aly AE, A. A. A. Zekry, and H. L. A. Zayed. "Matlab Code for LTE Convolutional Code and Viterbi Decoder." International Journal of Engineering Research & Technology (IJERT) 6.3 (2017): 578-581.

[27] Nanthini, S., et al. "An Efficient Low Power Convolutional Coding with Viterbi Decoding Using FSM." Asian Journal of Applied Science and Technology (AJAST) Volume 1 (2017).

[28] Rocha, Leandro MG, et al. "Physical implementation of an ASIC-oriented SRAM-based viterbi decoder." 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2017.

