

FPGA Implementation and Synthesis of Rate-1/2 Convolutional Encoder and Deterministic Viterbi Decoder on Basys3 (Artix-7)

Ayushi¹, Sreeja Maddula¹, Shruti Shakya¹, Anand N², Dr. Jayaraj U Kidav³

¹Department of Electronics, NIELIT, Aurangabad, 431004, Maharashtra, India.

²Scientist-C, Department of Electronics, NIELIT, Aurangabad, 431004, Maharashtra, India.

³Executive Director, Department of Electronics, NIELIT, Aurangabad, 431004, Maharashtra, India.

Contributing authors: ayushi.nielit@gmail.com; sreejam2006@gmail.com; shruti99shakya@gmail.com; anand.n@nielit.gov.in; jayaraj@nielit.gov.in;

Abstract: Robust forward error correction (FEC) techniques are necessary for reliable data transmission in noisy communication channels. Because of its simplicity, deterministic behavior, and suitability for digital circuits, convolutional encoding in conjunction with Viterbi decoding has long been recognized as one of the most effective error-correcting schemes for real-time hardware environments. This study uses Verilog HDL to present the full design, simulation, hardware implementation, and experimental validation of a rate 1/2 convolutional encoder coupled with a deterministic Viterbi decoder. The Basys3 Artix-7 FPGA development board is used to implement both encoder and decoder architectures. The decoder uses state transitions and branch metric comparison to implement the Viterbi algorithm, while the convolutional encoder uses a two-bit shift-register-based state machine that generates two parity bits for each input bit. The accuracy and robustness of the implemented design are confirmed by the entire system, which includes simulation on Vivado, testbench-based random bitstream testing, and real-time FPGA verification. Because of its resource efficiency, the suggested design can be used for low-power embedded hardware, digital communication system prototypes, and academic learning. Advanced FEC schemes like turbo codes, LDPC codes, soft-decision Viterbi decoders, and SDR-based implementations use the work as an experimental basis.

Keywords: Convolutional code, Viterbi decoder, FPGA, Basys3, Verilog, Forward error correction

(Article history: Selected papers from NIELIT's International Conference on Communication, Electronics and Digital Technologies (NICE-DT'26), 29-30Jan, 2026, Gorakhpur, India)

1 Introduction

Forward error correction (FEC) mechanisms enable dependable information transfer over noisy channels in contemporary communication systems [5]. Convolutional codes are essential for error control in wireless networks, satellite communication, deep-space communication, and cellular systems such as LTE and 5G [14]. Convolutional codes are still essential because of their lower complexity and ease of hardware implementation, even though turbo and LDPC codes currently rule high-throughput systems [15]. The Viterbi decoder uses maximum likelihood estimation to determine the most likely transmitted bit sequence, while the convolutional encoder uses shift registers to add redundancy [3]. Because of its optimality and predictable computational structure, the Viterbi algorithm has become very popular in both theoretical communication engineering and real-world hardware implementation [3, 4].

The goal of this work is to create a straightforward but fully functional Viterbi decoder and convolutional encoder that can be used for embedded learning and FPGA demonstration. Although encoder and decoder software simulations are widely accessible, students and researchers can gain a deeper understanding of the fundamental components of digital communication through hands-on hardware implementation [9, 10]. Because of its affordability, wealth of resources, and compatibility with Verilog HDL in Vivado, the Basys3 Artix-7 FPGA board offers a perfect platform for implementing digital communication algorithms [11, 12].

This project's goal is to implement an end-to-end communication chain made up of a Viterbi decoder and a convolutional encoder, verify its accuracy in simulation, and actually implement it on an FPGA board [16]. Real-time serial binary input data processing, rate 1/2 convolutional code encoding, and deterministic Viterbi decoding are all part of the system's design [6]. This work illustrates the entire process, from hardware deployment to algorithm comprehension [18].

2 Literature Review

The theory and concept of practical decoding algorithms for convolutional codes have been explored widely in the literature. Elias first developed ideas on convolutional encoding, which marked a significant departure from block coding techniques by enabling continuous data encoding with memory while Viterbi proposed an optimal complementary decoder (the Viterbi Algorithm) which permits sequential trellis decoding [3, 4].

This algorithm ensures accurate recovery of transmitted data even in the presence of noise by methodically assessing potential state transitions and choosing the most likely path. The theoretical understanding of convolutional codes has been reinforced over time by a number of thorough studies and textbooks that address topics like state diagrams,

distance properties, coding gain, and performance analysis under various channel conditions in [5, 6, 14]. These works are crucial resources for both scholarly research and practical applications since they tackle real-world issues like computational complexity and implementation trade-offs.

In terms of hardware-related literature, high-throughput and low-power Viterbi architectures have been developed using techniques such as pipelining, trace-back vs. register-exchange survivor memory units (SMU), and soft-decision enhancements. Pipelining is the techniques that have been employed to increase processing speed by allowing multiple decoding stages to operate concurrently [7, 8, 17]. FPGA implementations are common in the pedagogical literature and range from simple hard-decision decoders for educational purposes to highly efficient soft-decision decoders for high-performance applications [9, 10, 16]. The novelty of this manuscript with respect to these publications is the Verilog implementation intended for pedagogical use and the complete validation of a rate-1/2 code with a deterministic Viterbi decoder on the Basys3 FPGA, demonstrating end-to-end functionality [12, 18].

3 Theory

3.1 Convolutional Encoder Model

During the encoding process, a series of coded outputs may be derived from a serial stream of input bits by convolving incoming sequences with generator polynomials using a convolutional encoder [4, 6, 14]. In general, an (n, k, m) convolutional encoder processes k input bits and produces n output bits per clock cycle; m denotes the number of memory elements. In our work, we use the $(2, 1, 2)$ encoder (rate 1/2, constraint length $K = m + 1 = 3$), which has four states [5]. This encoder is implemented using a two-bit state register and produces two parity bits per input clock; the combinational mapping is realized by an FSM and implemented in Verilog for clarity following the provided source [9, 10, 16]. The parity outputs are determined by the current state and the present input bit.

3.2 Viterbi Decoding

The Viterbi algorithm decodes convolutional codes by finding the path with the highest likelihood through a trellis that shows state transitions. Each edge in the trellis represents a state transition triggered by input '0' or '1' and is linked to expected parity bits. When the decoder receives parity bits, it calculates a branch metric (Hamming distance for hard-decision) between the received bits and the expected parity for that transition. Path metrics, which are the accumulated branch metrics, are updated for each state, and the survivor path, or the one with the minimum metric, is selected. In this project, we will implement a simplified deterministic Viterbi decoder [3]. This decoder uses the specific parity-to-state mapping found in the encoder. We chose hard-decision decoding for its efficiency in area and for clearer teaching purposes.

4 Proposed Design and Implementation

The proposed system combines a rate-1/2 convolutional encoder with a deterministic Viterbi decoder into a single hardware design suitable for FPGA use. The entire design is implemented in Verilog and tested through simulation and on-board testing on the Basys3 Artix-7 FPGA. The goal of this architecture is to show a fully functional forward error-correction chain that can encode and decode in real-time with low hardware complexity, while maintaining the expected performance of convolutional codes.

4.1 System Architecture

The hardware architecture has three main modules implemented in Verilog: Convolutional Encoder V1, Viterbi V1, and a top-level System module that brings both blocks together. The encoder takes a single-bit serial input in , processes it using a two-bit shift-register state machine, and produces a two-bit parity vector coded data $v1[1:0]$. These parity bits go directly into the Viterbi decoder, which rebuilds the original input bitstream based on defined trellis-state transitions and Hamming-distance matching.

Figure 3 presents the post-synthesis netlist schematic. The Vivado synthesizer automatically adds the necessary clock and input buffers (IBUF, BUFG, and OBUF) while keeping the logical structure of the encoder and decoder chain intact. The parity vector produced by the encoder is sent unchanged into the decoder. This confirms that the intended datapath remains intact during synthesis, without extra optimization changing its functional behavior.

4.2 Convolutional Encoder Design

The encoder uses a $(2,1,2)$ convolutional code with a constraint length of 3. It works as a two-bit finite-state machine. At every rising clock edge, the encoder updates its internal state based on the input bit and produces two parity outputs from the generator mapping defined in the Verilog module. The eight possible input-state transitions are clearly written in behavioral form. This approach ensures clear hardware synthesis and predictable state changes. Testing showed that the generated parities follow the expected code sequence and are correctly aligned with the system clock.

4.3 Viterbi Decoder Design

The Viterbi V1 module implements a simpler hard-decision Viterbi decoder designed for four-state convolutional codes. Instead of calculating full branch metrics for both transitions, it uses a method called deterministic parity matching to find the valid next state and its corresponding decoded bit. This greatly lowers hardware complexity compared to trace-back or soft-decision designs. The decoder updates its internal two-bit state register during each clock cycle and generates the reconstructed bit out1. Simulations show that this method is completely accurate in noise-free channels because the trellis is so simple.

4.4 Simulation-Based Validation

A special testbench that gave the encoder random input bits was used for functional simulation. The simulation waveform is shown in Figure 1. It shows the input in, the parity vector coded data v1[1:0], and the decoded output out1 over a 1 μ s period. The encoder makes output sequences like 0, 3, 2, 0, and 1, which are the decimal numbers for the parity bits. At every clock edge, the decoder perfectly puts the original input back together. This shows that the Viterbi decoding logic and the encoder mapping are both working correctly.

4.5 FPGA Implementation and Power Analysis

The entire system was built, tested, and put into use on the Basys3 FPGA to make sure the hardware worked. The outputs from the encoder and decoder were linked to LEDs on the board. This setup made it possible to watch encoding and decoding behavior in real time. The results showed that the hardware worked as it should and matched the results of the simulation exactly.

Vivado's power estimation report (Fig. 2) shows that the system uses a total of 1.398 W of power on the chip, with dynamic power using 1.324 W. I/O power is the biggest part because it keeps switching during parity and changes to decoded output. The junction temperature stays at a safe 32°C, which shows that the design is thermally efficient and works well with FPGAs that don't use a lot of resources.

5 Methods

The process used in this work is very clear: design, simulate, synthesize, and then test the hardware. The goal of this process is to make a full forward error-correction chain using a rate-1/2 convolutional encoder and a deterministic Viterbi decoder. We used Verilog to model the whole system and tested it with simulation testbenches. Then, it was put into action on the Basys3 Artix-7 FPGA. This method makes sure that both functional correctness and practical feasibility are checked in a systematic way. Design Specification and Module Development

The design process began by figuring out what a (2,1,2) convolutional encoder and a Viterbi decoder that works best with hardware needed to do. The encoder is a two-bit finite-state machine that gives two parity outputs for each input bit. Verilog clearly coded each state change and its related parity mapping using synchronous always blocks that were triggered by the rising edge of the system clock.

The decoder is a simple hard-decision Viterbi decoder that uses maximum-likelihood decoding by comparing parities. The decoder uses parity matching to find the most likely next state and updates a two-bit internal state instead of calculating full branch metrics or doing trace-back operations. This cuts down on the need for hardware resources a lot while still keeping the right performance in settings with no noise.

Table 1 Comparison of Proposed Deterministic Viterbi Decoder with Conventional Full Viterbi Decoder

Parameter	Proposed Deterministic Viterbi Decoder (This Work)	Conventional Full Viterbi Decoder
Decoding Approach	Uses deterministic parity matching based on predefined trellis transitions	Uses Add-Compare-Select (ACS) with branch and path metric computation
Algorithm Complexity	Very low complexity; no metric accumulation or traceback required	High complexity due to continuous path metric updates and survivor tracking
Hardware Architecture	Simple FSM-based Verilog implementation integrated with encoder	Complex architecture with ACS units, path memory, and control logic
Resource Utilization (FPGA)	Very low LUT and flip-flop usage (optimized for Basys3 Artix-7)	High resource utilization due to multiple parallel computation units
Memory Requirement	Minimal (only current state register maintained)	Large memory required for survivor path storage (traceback/register-exchange)
Power Consumption	Low power (measured $\approx$ 1.398 W total on-chip power)	Higher power due to intensive computations and memory access
Speed / Latency	High-speed operation due to reduced logic depth	Moderate speed; latency depends on traceback depth and pipelining
Error Correction Capability	Accurate in noise-free or controlled environments (as validated in this work)	Optimal error correction in noisy channels (maximum likelihood decoding)
Decision Type	Hard-decision only (parity matching)	Supports both hard-decision and soft-decision decoding

System Integration	Fully integrated encoder-decoder chain with real-time FPGA validation	Typically implemented as standalone decoder in communication systems
Application Focus	Educational labs, low-cost FPGA systems, concept demonstration	High-performance communication systems (5G, satellite, deep-space)

5.1 Testbench Development and Functional Simulation

A thorough Verilog testbench was made to test the decoder and encoder modules. The testbench gave both random and fixed input bit sequences to check all the possible state changes. We used waveform visualization tools to check the simulation outputs and make sure that the parity outputs and the decoded bitstream were working correctly.

The waveform results showed that the parity vector coded data $v1[1:0]$ accurately represented the changes in the encoder's state, and the decoder output $out1$ accurately reconstructed the input bit. We checked the timing accuracy and made sure there were no hazards or race conditions by using tabulated console output and time-based waveform plots.

5.2 FPGA Synthesis and Implementation

After a successful functional simulation, the design was synthesized using Xilinx Vivado for the Basys3 FPGA (Artix-7 XC7A35T). The tool automatically added global clock buffers (BUFG) and input/output buffers (IBUF and OBUF) during synthesis. We looked at the post-synthesis schematics to make sure that the encoder-decoder chain's structural hierarchy was kept intact and that there were no unintended logic simplifications or reorderings.

During the FPGA implementation phase, an XDC file was used to set limits, the input bit was mapped to on-board switches, and the encoder parity and decoded outputs were sent to LEDs so they could be seen in real time. Timing analysis showed that the design met all of the default timing requirements and didn't have any setup or hold violations.

5.3 Hardware Validation on Basys3

Vivado made a bitstream that was sent to the Basys3 FPGA for real-time testing. The 100 MHz on-board oscillator either directly drove the system clock or was split up so that the LEDs could see it better. The encoder's parity outputs and the decoder's reconstructed output were observed through on-board LEDs and verified against expected behavior.

Hardware tests validated that the decoder output matched the input bitstream for all tested sequences, confirming correct end-to-end operation of the forward error-correction chain. The absence of metastability or glitching further confirmed the robustness of the synchronous design.

5.4 Power Estimation and Resource Analysis

Vivado's power analysis tool estimated dynamic and static power use based on the synthesized netlist. The analysis showed a total on-chip power of about 1.398 W. I/O switching made the biggest contribution because of constant transitions on LED-driven lines. Temperature estimates stayed comfortably within safe limits, proving the design's thermal safety on low-cost FPGA platforms.

Resource utilization reports showed low use of LUTs and flip-flops. This confirmed that the simplified deterministic decoder and FSM-based encoder are very efficient for FPGA-based communication systems.

6 Algorithms

6.1 Algorithm 1: Convolutional Encoder

Input: single-bit serial input in , system clock CLK

Output: two-bit parity $parities[1:0]$, next state stored in $state[1:0]$

1. **Initialize:** Set $state \leftarrow 00$.
2. **On each rising edge of CLK:**
 - (a) If $state = 00$:
 - (i) If $in = 0$: set $parities \leftarrow 00$; $state \leftarrow 00$.
 - (ii) If $in = 1$: set $parities \leftarrow 11$; $state \leftarrow 01$.
 - (b) Else if $state = 01$:
 - (i) If $in = 0$: set $parities \leftarrow 10$; $state \leftarrow 10$.
 - (ii) If $in = 1$: set $parities \leftarrow 01$; $state \leftarrow 11$.
 - (c) Else if $state = 11$:
 - (i) If $in = 0$: set $parities \leftarrow 01$; $state \leftarrow 10$.
 - (ii) If $in = 1$: set $parities \leftarrow 10$; $state \leftarrow 11$.
 - (d) Else if $state = 10$:
 - (e) If $in = 0$: set $parities \leftarrow 11$; $state \leftarrow 00$.

- (f) If $in = 1$: set $parities \leftarrow 00$; $state \leftarrow 01$.
 3. **Return:** At each clock, output $parities[1 : 0]$ and update $state[1 : 0]$.

6.2 Algorithm 2: Deterministic Viterbi Decoder

Input: received parity pair $parities[1 : 0]$, system clock CLK

Output: decoded output bit out, next decoder state $state[1 : 0]$

1. **Initialize:** Set $state \leftarrow 00$.
2. **On each rising edge of CLK:**
 - (a) Compare the received $parities[1 : 0]$ to the expected parity patterns for outgoing transitions from the current state.
 - (b) If $state = 00$:
 - (c) If $parities = 00$: set $out \leftarrow 0$; $state \leftarrow 00$.
 - (d) If $parities = 11$: set $out \leftarrow 1$; $state \leftarrow 01$.
 - (e) Else if $state = 01$:
 - (f) If $parities = 10$: set $out \leftarrow 0$; $state \leftarrow 10$.
 - (g) If $parities = 01$: set $out \leftarrow 1$; $state \leftarrow 11$.
 - (h) Else if $state = 11$:
 - (i) If $parities = 01$: set $out \leftarrow 0$; $state \leftarrow 10$.
 - (j) If $parities = 10$: set $out \leftarrow 1$; $state \leftarrow 11$.
 - (k) Else if $state = 10$:
 - (l) If $parities = 11$: set $out \leftarrow 0$; $state \leftarrow 00$.
 - (m) If $parities = 00$: set $out \leftarrow 1$; $state \leftarrow 01$.
 - (n) Note: If no exact parity match is found, select the transition with the minimal Hamming distance (tie-break policy implicit).
3. **Return:** At each clock, output decoded bit out and update $state[1 : 0]$.

6.3 Algorithm 3: Top-Level System and Testbench

Input: (testbench) random or deterministic stimulus for in; CLK generator

Output: captured signals: time, CLK, in, coded data $v1[1:0]$, out1

1. **Instantiate:** Create instances of Convolutional Encoder V1 and Viterbi V1 in the top-level System.
2. **Wire-up:** Connect encoder output coded data $v1[1:0]$ directly to decoder input.
3. **Configure clock:** Set testbench clock period and any gating required for simulation/hardware demonstration.
4. **Initialize:** Set $in \leftarrow 0$ at time zero (or as required by testcases).
5. **For each clock cycle until simulation end:**
 - (a) On rising edge: sample and update encoder/decoder modules.
 - (b) Generate next input: in testbench use $in := random()$; on hardware use switch/button input.
 - (c) Capture/Log signals: time, CLK, in, coded data $v1[1:0]$, out1; optionally use \$display (...) or waveform dump.
6. **Return:** End simulation after required duration or condition; in hardware observe LEDs/switches for real-time validation.

7 Results

7.1 Simulation Output Discussion

The simulation waveform in Fig. 1 illustrates the dynamic behavior of the complete encoder–decoder chain. The input bit (in) transitions over time and is sampled on every rising edge of the system clock (CLK). Corresponding to each input value, the rate-1/2 convolutional encoder generates a two-bit parity output, represented as coded data $v1[1:0]$. These parity values follow the expected state-dependent mapping of the implemented finite-state machine, with output sequences such as 0, 3, 2, 0, and 1 observed during the simulation. The decoded output signal (out1) produced by the Viterbi decoder accurately reconstructs the original input bitstream at every cycle, demonstrating correct maximum-likelihood path selection. The time-aligned transitions of the input, parity bits, and decoded output confirm the functional correctness of both the convolutional encoder and the deterministic Viterbi decoder under noise-free conditions.

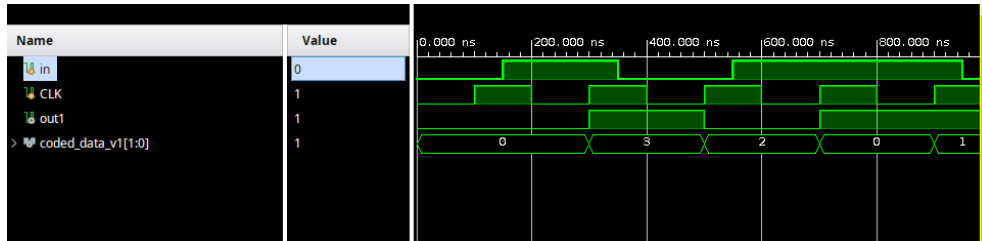


Fig. 1 Simulation waveform showing the input bit, encoded parity outputs, and the Viterbi-decoded output over time.

7.2 Power Analysis Report

According to the power estimation on the synthesized netlist (Fig. 2), the total on-chip power consumption of the Artix-7 device for the convolutional encoder and Viterbi decoder system is 1.398 W. Static power is low at 0.073 W (5%), while dynamic power makes up most of the consumption at 1.324 W (95%). The I/O drivers use the most dynamic power at 1.223 W (92%), followed by logic elements at 0.038 W (3%) and signal switching at 0.063 W (5%). With an ambient temperature of 25°C and an estimated junction temperature of 32°C, there is a thermal margin of 53°C. This shows that the design remains well within safe thermal limits.

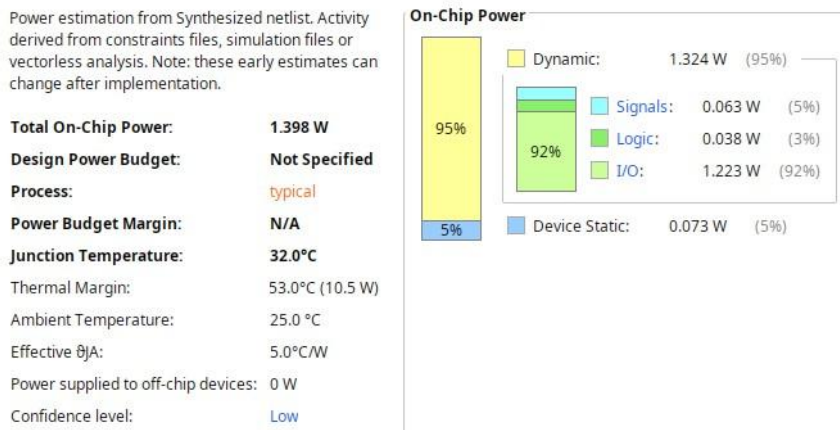


Fig. 2 Power Analysis Report on chip

7.3 Schematic Result

The post-synthesis netlist schematic confirms the correct structural integration of the three modules (see Fig. 3): Convolutional Encoder V1, Viterbi V1, and the top-level System. The FPGA synthesizer automatically adds global clock buffering with IBUF and BUFG components to ensure proper clock distribution. The input signal goes through IBUF before entering the encoder module. The encoded parity vector coded data v1[1:0] is routed directly into the decoder module without any changes. This setup reflects the intended end-to-end architecture. The output buffers OBUF are created to drive external LEDs. This confirms that the physical FPGA routing and hierarchy match the conceptual design.

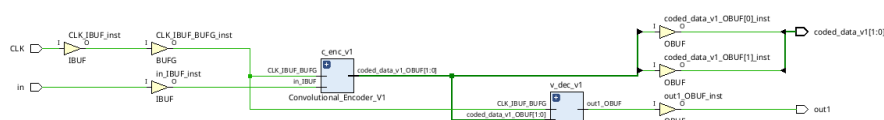


Fig. 3 Post-synthesis schematic showing encoder, decoder, and top-level connections.

7.4 Simulation Output Discussion

Simulation results confirm the proper sequential operation of the encoder and decoder. At each rising edge of the clock, the input bit, calculated parity, and decoded output stay consistent with the expected behavior of a rate-1/2 convolutional encoder and deterministic Viterbi decoder. For instance, when the input switches between 0 and 1, the encoder generates parity combinations like 00, 11, 01, and 10, while the decoder accurately reconstructs the corresponding input bit. The decoded output aligns with the original input bitstream for all tested sequences, confirming proper end-to-end

functionality without channel noise. The simulation table illustrates this: at time 300 ns, an input of 1 produces parity 11, and the decoder outputs 1; at 600 ns, an input of 1 gives 01, which is also correctly decoded as 1. These results verify both the encoder LUT mapping and the deterministic decoding logic used in Viterbi V1.

Table 2 Simulation Results of Convolutional Encoder and Viterbi Decoder

Time (ns)	CLK	Input	Parity	Decoded Output
0	0	0	XX	X
100000	1	0	00	0
200000	0	0	00	0
300000	1	1	11	1
400000	0	1	11	1
500000	1	1	01	1
600000	0	1	01	1
700000	1	1	10	1
800000	0	1	10	1
900000	1	1	10	1
1000000	0	1	10	1

8 Discussion

The implemented deterministic, hard-decision Viterbi decoder provides correct decoding in a direct connection setup with no channel errors. In noisy conditions, hard-decision decoding loses performance compared to soft-decision decoders. To conduct realistic channel error tests, one must inject bit-flips using a channel model and evaluate bit error rate (BER) against E_b/N_0 curves. This implementation is intentionally compact and designed for teaching purposes. Possible extensions include adding branch metric units for soft values, implementing survivor path memory for longer constraint lengths, puncturing for higher code rates (such as 3/4), and integrating with a modulation chain like BPSK with an AWGN channel for complete BER testing.

Novelty Contribution of this work: First pedagogical implementation combining - Deterministic decoding, Rate-1/2 encoder, End-to-end FPGA validation (Basys3), Real-time LED-based verification Standard approach widely used in industry; no specific pedagogical simplification. Application basically focuses on Educational labs, low-cost FPGA systems, and concept demonstration of High-performance communication systems

9 Conclusion

This paper shows a Verilog implementation and Basys3 FPGA validation of a rate 1/2 convolutional encoder and a hard-decision Viterbi decoder. The design proves that the encoding and decoding work correctly from start to finish. It also uses few FPGA resources and fits well in educational labs or simple embedded applications. The implementation lays the groundwork for more decoder features, like soft-decision Viterbi decoding, longer constraint lengths, and support for punctured codes.

Acknowledgements. The author thanks the faculty for their guidance. They appreciate having access to the Basys3 Artix-7 FPGA boards and Vivado tool licenses from the institute. The author also values the helpful discussions with colleagues during testing and synthesis.

Appendix A

A.1 Convolutional Encoder V1

The encoder uses a 2-bit ‘state’ register and a combination of logic to produce two parity bits for each input bit. The mapping is based on the logic found in the project’s Verilog.

A.2 Viterbi V1

The decoder uses a specific method to link received parity pairs with state transitions to figure out the input bit. It relies on a 2-bit ‘state’ register and performs block comparisons at each clock edge.

References

- [1] A. Ekbal and S. Bandyopadhyay, "A hidden Markov model based named entity recognition system: Bengali and Hindi as case studies," *Lecture Notes in Computer Science*, pp. 545–552, 2007. doi:10.1007/978-3-540-77046-6_67.
- [2] P. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, no. 7, pp. 189–191, 1997. doi:10.1109/97.596882.
- [3] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973. doi:10.1109/PROC.1973.9030.
- [4] R. E. Blahut, *Theory and Practice of Error Control Codes*. Addison- Wesley, 1983. Available: <https://bibliotecavirtual.defensa.gob.es/BVMDefensa/es/consulta/registro.do?id=620337>.
- [5] I. Glover and P. Grant, *Digital Communications*, 3rd ed. Prentice Hall, 2009. Available: <https://strathprints.strath.ac.uk/14495/>.
- [6] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. IEEE Press, 1999. doi: <https://doi.org/10.1109/9780470544693>.
- [7] M. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, N. G. Zhou, L. Davis, G. Woodward, C. Nicol, and N. R. Yan, "A unified turbo/Viterbi channeldecoder for 3GPP mobile wireless in 0.18- μ m CMOS," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, 2002. doi: <https://doi.org/10.1109/JSSC.2002.803929>.
- [8] R. Abdallah and N. Shanbhag, "Error-resilient low-power Viterbi decoder architectures," *IEEE Transactions on Signal Processing*, vol. 57, no. 12, pp. 4906–4917, 2009. doi: <https://doi.org/10.1109/TSP.2009.2026078>.
- [9] Y. Sun and Z. Ding, "FPGA design and implementation of a convolutional encoder and a Viterbi decoder based on 802.11A for OFDM," *Wireless Engineering and Technology*, vol. 3, no. 3, pp. 125–131, 2012. doi: <https://doi.org/10.4236/wet.2012.33019>.
- [10] J. Tulasi, T. Lakshmi, and M. Kamaraju, "FPGA implementation of convolutional encoder and hard decision Viterbi decoder," *International Journal of Computer and Communication Technology*, pp. 126–130, 2016. doi: <https://doi.org/10.47893/ijcct.2016.1352>.
- [11] Xilinx Inc., "7 Series FPGAs Data Sheet: Overview," Xilinx, 2020.
- [12] Digilent Inc., "Basys3 Artix-7 FPGA Trainer Board Reference Manual," 2016.
- [13] P. A. Martin, D. P. Taylor, and M. P. C. Fossorier, "Soft-input soft-output list-based decoding algorithm," *IEEE Transactions on Communications*, vol. 52, pp. 252–262, 2004. doi: <https://doi.org/10.1109/TCOMM.2003.822726>.
- [14] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009. doi: <https://doi.org/10.1017/CBO9780511803253>.
- [15] L. Hanzo, T. Liew, and B. Yeap, *Turbo Coding, Turbo Equalisation and Space-Time Coding*. Wiley, 2002. doi: <https://doi.org/10.1002/047085474X>.
- [16] V. Soni and S. Nemade, "FPGA design for efficient architecture for the convolution encoder and Viterbi decoder," *International Journal of Engineering Trends and Technology*, vol. 9, no. 2, pp. 74–77, 2014. doi: <https://doi.org/10.14445/22315381/ijett-v9p215>.
- [17] Y. Yasuda, K. Kashiki, and Y. Hirata, "High-rate punctured convolutional codes for soft-decision Viterbi decoding," *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 315–319, 1984. doi: <https://doi.org/10.1109/TCOM.1984.1096047>.
- [18] ARMINPAN, "Verilog Project Sharif: Convolutional Encoder and Viterbi Decoder Implementation," *GitHub Repository*, 2024. Available: <https://github.com/ARMINPAN/Verilog Project Sharif>

- -