Journal of Advanced Engineering Trends
ISSN : 2682 - 2091

Vol. 44, No. 1, January 2025

Development and Implementation of pipeline Convolutional Coding using FPGA

Sara M. Hassan[1,*], Aziza I. Hussein[2], Ashraf A. Khalaf[3]
[1] *Electronics and Communications Engineering Dep., Modern Academy for Engineering and Technology, Cairo, Egypt*
[2] *Electrical and Computer Engineering Dep., Effat University, Jeddah, KSA*
[3] *Electrical Engineering Dep., Faculty of Engineering, Minia University, Minia, Egypt*
** Corresponding author(s) E-mail: sara.hassan@eng.modern-academy.edu.eg*

ARTICLE INFO

Article history:
Received: 9 June 2024
Accepted: 17 September 2024
Online: 10 March 2025

Keywords:
Convolutional coding,
FPGA,
Forward error correction,
Viterbi decoding,
VHDL

ABSTRACT

Channel coding is essential for ensuring reliable data transmission in challenging wireless communications. Improving spectrum efficiency involves leveraging efficient forward error correction (FEC) methods. Viterbi decoding plays a critical role in Convolutional channel coding for accurate error detection and correction, particularly in LTE and Satellite communication systems. This article discusses the simulation and FPGA implementation of a newly proposed non-systematic Convolutional system featuring a block interleaver and 64-QAM Mapping under AWGN and Rayleigh channel conditions. The system adopts a Convolutional coding rate equal to 1/3 and a constraint length of 7, utilizing a Trellis-diagram for encoding and the Viterbi-algorithm for decoding with hard decision decoding. Additionally, a pipeline coding approach is employed. Simulations are conducted using MATLAB-R2023b, and the implementation is executed on Virtex 6 (XC6VLX240T) FPGA using Xilinx 14.7. The study reveals that the pipeline technique demands more FPGA resources compared to traditional methods while still utilizing a small resource block from Virtex 6, with 3% and 9% usage of slice registers and LUTs, respectively. Moreover, the system's timing is reduced from 24 to 14 clock cycles, enhancing the efficiency of entirely LUT-FF pairs from 55% to 63%.

1. Introduction

In modern digital communications systems, particularly in satellite communications; 4G and 5G mobile generation, Forward Error Correction (FEC) is becoming more well acknowledged for its advantages and prospective benefits. As a result, channel coding is increasingly important in the design of these systems [1]. The primary difficulty with FEC systems usually lies in the real development of strong decoders that will work with large amounts of information, provide less mistake probability while remaining effective and not overly hard to accomplish. The difficulty of engineering and producing rapid encoding and decoding equipment at the cheapest feasible budget has been exacerbated when information rates grow, extending into hundreds of several mega-bits for each second (Mbits/s) as well as greater. [2-6].

Convolutional encoding is commonly employed in order to encode information through FEC methods. The upcoming decoding was carried out via Viterbi decoding. Convolutional encoding and Viterbi decoding are especially well-suited for wireless channels. The signal being transmitted is primarily distorted by additive white Gaussian noise. Convolutional codes are frequently preferred over block codes in practice since they were one of the first codes for which algorithms were devised [7-9]. Block codes accept discrete blocks of k-symbols and output blocks of n-symbols that only rely on the k input symbols. Convolutional codes are commonly referred to as streaming codes because they frequently work on a continuous flow of symbols rather than separate messaging chunks. Those have remaining rates (R = k/n codes), which accept k fresh symbols at every time period and generate n brand-new symbols [10-11]. This research introduces a design and implementation of digital communication kit, modeling the error detection and repair mechanisms in digital data sent by AWGN and Rayleigh. The Rayleigh channel impact was considered since fading channels are useful approximations of real-world occurrences in wireless communication, and there were no burst defects in the AWGN scenario to assess the advantages of block interleavers [12]. Gaussian white noise is an assortment of additive noise that is typically encountered in electrical circuits and has a Gaussian distribution with all frequencies in existence. It is characterized by a zero mean and variance parameter, which specifies the amount of noise in the input. The high speed and pipelining properties of the FPGA processor, together with pipeline coding, have been used to reduce processing time and hence boost data throughput. Previous studies offered some strategies to expedite and increase the error detection and repair processing for convolutional coding using MATLAB simulation but did not introduce or provide any real-time implementation strategies. [13–16]. However, the pipeline technique has the advantage of introducing more time response while maintaining acceptable system complexity and eliminating the necessity for mathematical operations reduction. [17] used a parallel Viterbi MATLAB simulation of the AWGN channel effect. While [18] produced the pipeline VLSI for the Viterbi decoder, [19] built another Viterbi decoder architecture employing an ASIC application.

This article provides the MATLAB simulation and FPGA implementation of the suggested non-systematic Convolutional system with block interleaver and 64-QAM mapping over the AWGN and Rayleigh channels impact. The suggested

convolutional coding system parameters are equivalent to those employed for LTE prior to turbo-coding [20-21]. The Convolutional coding rate (k/n = 1/3) and constraint length is (L = 7). The encoder applies a trellis diagram, and the decoder employs the Viterbi algorithm with hard decision decoding. Pipeline implementations for convolutional coding are created, and their complexity is evaluated. The simulation was executed out using MATLAB (version R2023b). The implementation has been carried out making use of the Virtex 6 (XC6VLX240T) FPGA kit and VHDL codes. The VHDL codes were written using Xilinx package version 14.7. The results of the simulation were displayed via the ISim simulator. The originality of this article stems from the simulation of a newly proposed system that combines a non-systematic convolutional system with a block interleaver and 64-QAM mapping, specifically designed for operation under AWGN and Rayleigh channel conditions. Furthermore, the article includes the FPGA implementation of this proposed system, along with a comparison of results obtained from both the simulation and the implementation. The second section of the article is headed "The proposed system block diagram". Section three illustrates the system's FPGA use, while Section four discusses pipeline coding. Section five provides full system verification, whereas Section six offers the conclusions.

2. The Proposed System Block Diagram

This section offers a proposed block diagram for Convolutional Coding. The block diagram includes three phases in the transmitter and reverse actions at the receiver. The first stage of the transmitter is a convolutional encoder which employs a trellis diagram, while the Viterbi algorithm applies in the receiver. The second stage is a block interleaver, and the third stage is the Mapper. The Mapper output data is sent over the AWGN or Rayleigh channel. Fig. 1 illustrates the suggested convolutional channel coding block diagram.

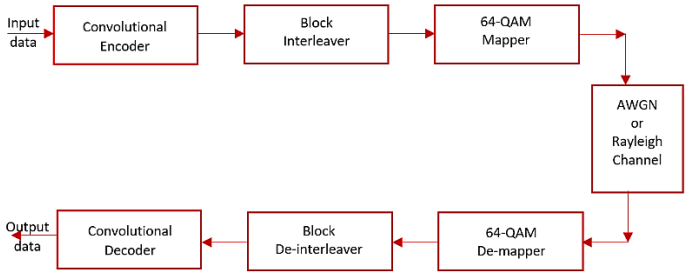


Figure 1: Convolutional channel coding block diagram

2.1. Convolutional coding

In Convolutional processing, information pieces arrive sequentially instead of in huge chunks. The label Convolutional description comes from the reality that redundant information is produced by modulo-2 convolutions at a convolutional encoder. The convolutional encoder can be thought of as a finite-state machine that includes an M-stage shift register, modulo-2 addition machines, and multiplexing devices. A convolutional processor has k entries and n outcomes; hence, its rate equals k/n. The makers to convolution-coding circuits frequently define the algorithms by employing quantities (n; k; L). An amount L is referred to as the code's constraint length, and it reflects the largest number of bit values in the single output flow might be impacted by any given input bit. Fig. 2 illustrates a non-systematic convolution encoder with an execution rate of 1/3. Equations (1, 2, and 3) can be used to represent the output response stream yi(n) of the rate 1/3 linearly convolution encoding.

$$y1(n) = x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-6) \quad (1)$$

$$y2(n) = x(n) + x(n-2) + x(n-3) + x(n-5) + x(n-6) \quad (2)$$

$$y3(n) = x(n) + x(n-1) + x(n-2) + x(n-4) + x(n-6) \quad (3)$$

Thus, the encoder's binary impulse responses are as follows: g1(n) = (1111001), g2(n) = (1011011), and g3(n) = (1110101). The constraint length "L" of a convolutional code is equal to the total length that defines the longest entry shifting register with the maximum number of storage elements in addition to one. m indicates the number of memory cells (shift register). Fig. 2 shows a rate of 1/3 for the non-systematic convolution encoding, which contains six memory cells. With m=6 and L = m + 1 = 7, the number of states is 2^m = 2^6 = 63. As a result, the corresponding state diagram of the trellis algorithm contains 63 transition states. Table 1 displays the state transition diagram for the assumed input sequence (1 0 1 0). Fig. 3 depicts the ISim simulation of convolutional encoder results when the reset equals zero. The memory cells are initialized with zeros and the current state (000000). When the reset is equal to one and the clock is high, the output sequence carried out as expected from state transitions as shown in Table 1. Fig. 4 depicts the convolutional encoder RTL schematic. The convolutional decoder by Viterbi algorithm is the reverse operations of encoder processing (backward direction on trellis). Fig. 5 illustrates the convolutional decoder RTL schematic. Fig. 6 depicts the Bit Error Rate (BER) performances of the convolutional encoder and decoder over the AWGN and Rayleigh channels performed with a hard decision. The chart additionally demonstrates the improvement in BER caused by Convolutional coding versus an un-coded system.

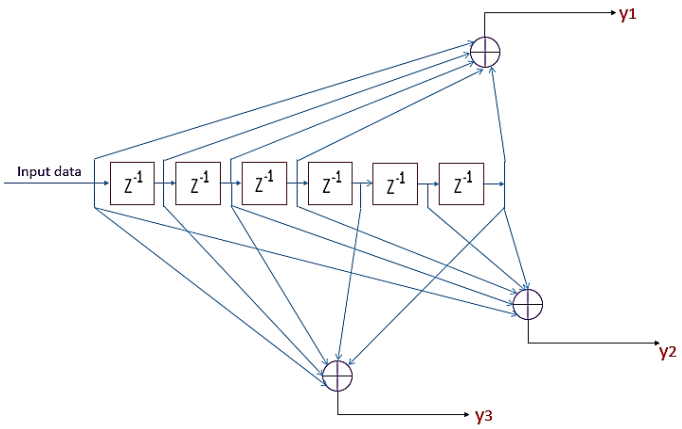


Figure 2: Rate 1/3 of the non-systematic Convolutional Encoder

Table 1: The state transitions diagram for assuming input sequence (1 0 1 0)

Input Sequence	1	0	1	0
Present State	S_0(000 000)	S_{32}(100 000)	S_{16}(010 000)	S_{40}(101 000)
Output Sequence	111	101	000	011
Next State	S_{32}(100 000)	S_{16}(010 000)	S_{40}(101 000)	S_{20}(010 100)

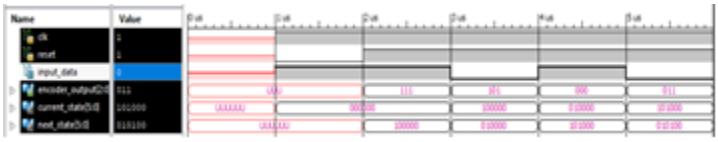


Figure 3: The ISim simulation of Convolutional Encoder results

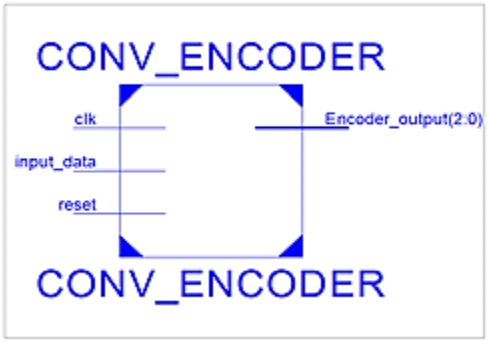


Figure 4: The RTL Schematic of Convolutional Encoder

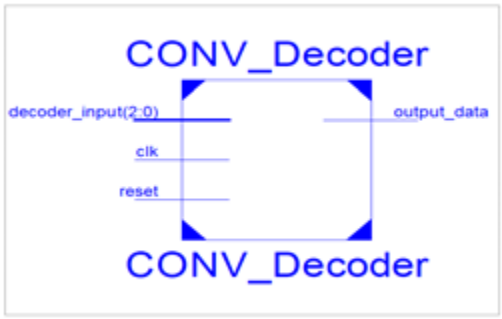


Figure 5: The Convolutional Decoder RTL Schematic

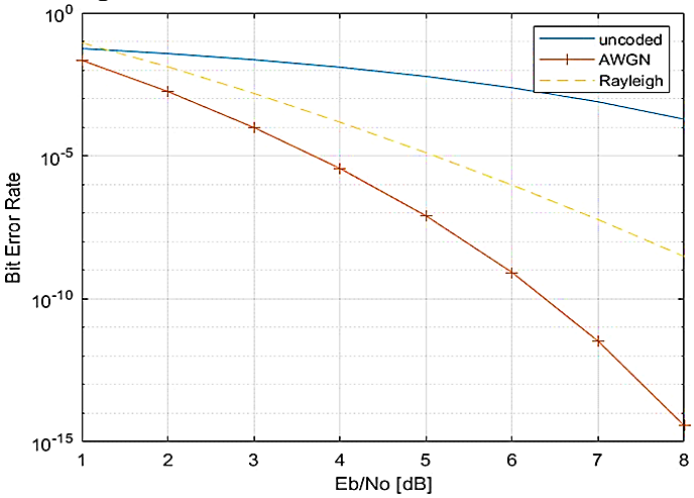


Figure 6: The BER performance of Convolutional Coding through an AWGN and Rayleigh channels simulated via MATLAB simulation

2.2. Interleaver

A block interleaver takes a set of symbols and rearranges them so that none of the symbols are repeated or omitted. For each particular interleaver, the number of symbols in each set remains constant. The convolutional interleaver is a data reordering technique that distributes bursts of mistakes and improves the efficiency of FEC algorithms in the presence of dropouts. The BER performance of convolutional can be considerably enhanced by using interleavers since they change the spacing attributes of the code through preventing low-weight code words. Block interleavers are an increasingly common type of interleaver in communication-systems. It populates the array by streaming the incoming information bit row by row before sending out the information column by column. Fig. 7 depicts a block interleaver the inputs are [01 … 100 … 1..… 0 …100 … 1] and the output is [00 … 00 … 1 … 0 … 1 … 01 … 1 1].

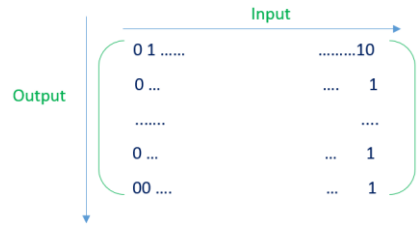


Figure 7: Block Interleaver write read ranking

The goal of bit-based interleaving is to optimize a system's diversity order while scattering burst faults caused by associated fading channels. Make the bit in relation to the transmission symbol not correlated or independent of one another. Fig. 8 demonstrates block interleaver operation. Fig. 9 displays the MATLAB results of the interleaver output with the entry from the Convolutional-Encoder output. The data that is entered (111 101 000 011) is the encoder trellis algorithm output, whereas the output of the Block Interleaver is (1100 1001 1101). Fig. 10 depicts the VHDL ISim simulation results of the block interleaver. As demonstrated in the figures, the ISim simulation findings are similar to the MATLAB simulation outcomes.

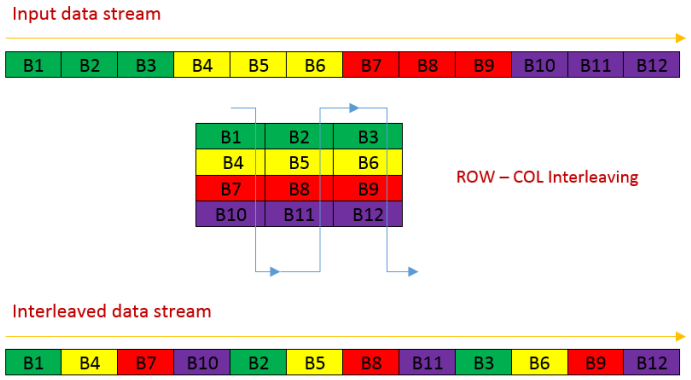


Figure 8: Block Interleaver operation

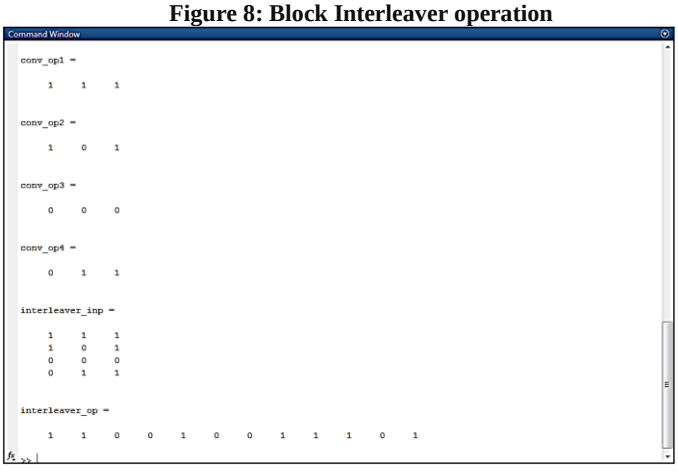


Figure 9: The block interleaver results by MATLAB command window

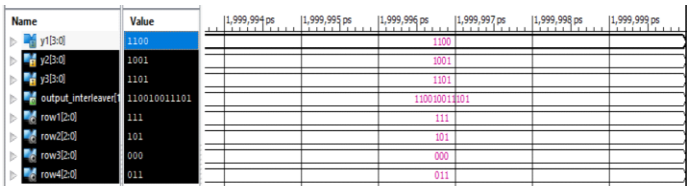


Figure 10: The ISim simulation results of block interleaver

2.3. Mapper

The Mapper with type M-QAM (M=64) is represented by the digital modulation constellation diagram. The Baseband symbols assign the outcome data acquired from the block interleaver to the Mapper input. That involves the I and Q components. The in-phase component is denoted as I, whereas the quadrature component is represented by Q. Fig. 11 illustrates the binary scatter plot. Fig. 12 displays the constellation diagram for the 64-QAM Mapper. Fig. 13 demonstrates the results of the DE-Mapper block's MATLAB simulation. While Fig. 14 displays the VHDL implementation Mapper RTL Schematic, (data_out) denoted the concatenated real and imaginary mapped data. Fig. 15 depicts the ISim simulator mapping results with the input sequence (1100 1001 1101) representing the preceding stage's output data (block Interleaver).

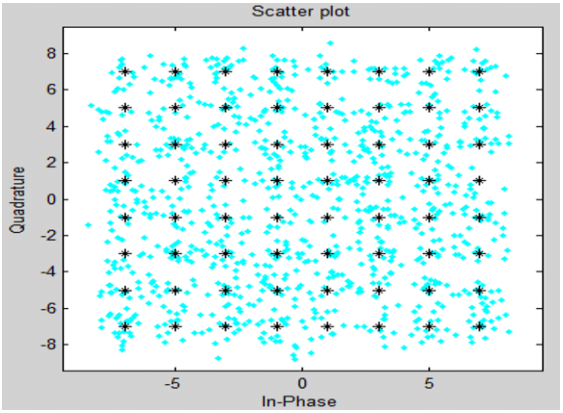


Figure 11: The scatter plot for the 64-QAM Mapper

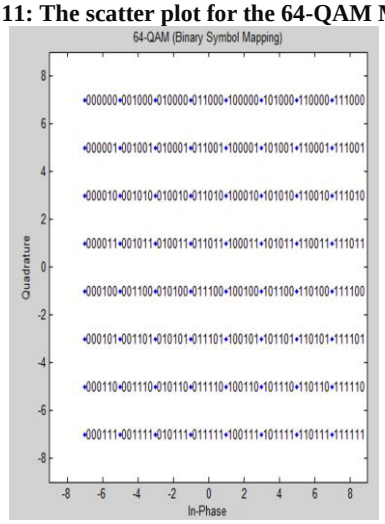


Figure 12: The binary constellation diagram for the 64-QAM Mapper

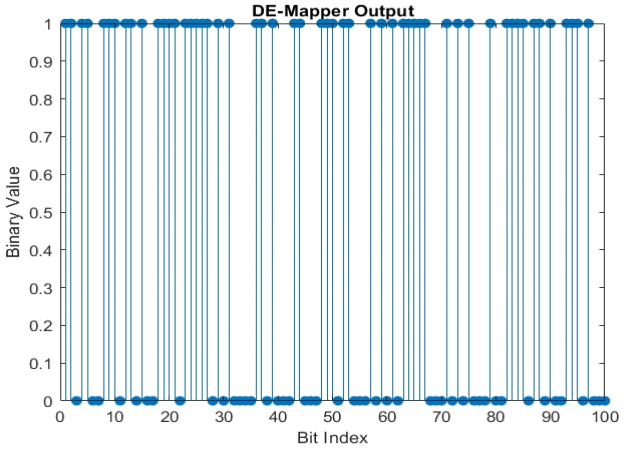


Figure 13: Simulation results for the DE-Mapper block

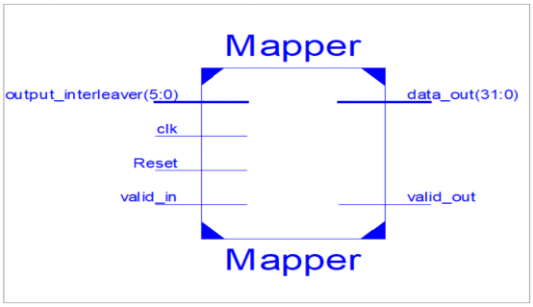


Figure 14: The RTL Schematic for Mapper

Figure 15: Mapping findings from the ISim simulator

3. FPGA UTILIZATION

The real-time implementation in this section is accomplished using the Virtex 6 (XC6VLX240T) FPGA kit. Fig. 16 illustrates an FPGA kit in usage. Table 2 illustrates the Virtex 6's overall system resource utilization. It is evident that the proposed convolutional coding technique makes use of only a fraction of Virtex 6's resource blocks. As seen in the table, the amount of used slice registers and look-up tables (LUTs) are extremely low, at 1% and 2%, respectively. There are a large number of totally LUT-FF pairs (55% utilization efficiency).

Figure 16. FPGA Kit for Virtex 6 (XC6VLX240T)

Table 2: The Virtex 6 whole system utilization

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	4002	301440	1%
Number of Slice LUTs	3747	150720	2%
Number of fully used LUT-FF pairs	2755	4994	55%
Number of bonded IOBs	47	600	7%
Number of Block RAM/FIFO	10	416	2%
Number of BUFG/BUFGCTRLs	1	32	3%

4. Pipeline Coding

The Trellis diagram performs convolution coding at the transmitter, and the same diagram, with 63 transition states in this suggested system scenario, is carried out at the receiver in a back-direction process known as the Viterbi Decoding method. In this section, the Encoding Diagram and Decoding Viterbi algorithm operations are performed using pipeline processing. The Viterbi algorithm works by following a coded-sequence {s0, s1, ...} or its signal-mapped counterpart {x0, x1, ...}, which equivalent to the path across the encoder-trellis. As a result of channel noise, the receiving sequence r might not have been perfectly equivalent to the path across a trellis. The algorithm for decoding identifies the path across the trellis that is most similar to the incoming sequence, where nearest is measured using the probability functioning proper for the channel. In context of the usual procedure, the next decoding phase operation does not begin until all of the previous path operations have been completed. As a result, this behavior, which might occur in the regular Viterbi decoding method, requires a long processing time. The pipelined method is an effective solution for the preceding problem since each clock initiates a new phase of the decoding procedure. At the end of the algorithm's clock cycles, the final surviving path with the fewest errors appears; This is the path that comes closest to the incoming sequence. Fig. 17 illustrates the pipeline processing concept. Table 3 demonstrates the system FPGA resource use with the pipelined Coding method. By comparing the pipeline utilization with usual Coding algorithm utilization in Table 2, it is evident that the pipeline method uses more kit logic components which increases the system's complexity. Although the pipeline technique requires more FPGA resources, it still uses a small resource block from Virtex 6. As indicated by the table, the number of slice registers and LUTs utilized are 3% and 9%, respectively. On the other hand, the pipelined processing reduces the system timing from 24 to 14 clock cycles, resulting in greater speed. Remarkably, the Pipeline strategy increased the percentage of entirely LUT-FF pairs (utilization effectiveness) from 55% in the traditional coding method to 63%. When compared to the ASIC implementation for the Viterbi decoder in [19], the proposed pipeline system consumed more FPGA resources. However, this is normal because this paper provides the pipeline technique for the encoder and decoder. Additionally, introduce more LUT-FF pairs for increased efficiency and greater speed. The pipeline strategy aims to boost design speed. Figure 18 depicts the time report for the normal technique, while Figure 19 depicts the timing report for the pipeline method. According to reports, the pipeline technique saves time while increasing design frequency when compared to the two previous figures. The Xilinx Suite package generates timing data automatically following the Synthesize procedure at the Xilinx timing closure. Timing closure is the process of verifying that an FPGA design meets all the necessary timing restrictions. These constraints establish the maximum allowable delay between a circuit's input and output, and they are critical to the design's proper operation.

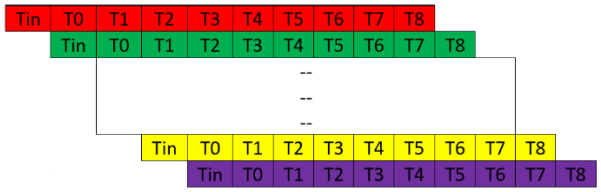


Figure 17: The Pipeline processing idea

Table 3 The Virtex 6 utilization of pipelined convolutional coding system

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	10615	301440	3%
Number of Slice LUTs	14344	150720	9%
Number of fully used LUT-FF pairs	9723	15236	63%
Number of bonded IOBs	85	600	14%
Number of BUFG/BUFGCTRLs	2	32	6%

Timing Summary:

Speed Grade: -1

Minimum period: 5.025ns (Maximum Frequency: 198.991MHz)

Minimum input arrival time before clock: 3.010ns

Maximum output required time after clock: 3.035ns

Maximum combinational path delay: 3.080ns

==========

Process "Synthesize - XST" completed successfully

Figure 18: Timing report for the normal method

Timing Summary:

Speed Grade: -1

Minimum period: 2.863ns (Maximum Frequency: 349.261MHz)

Minimum input arrival time before clock: 2.191ns

Maximum output required time after clock: 5.437ns

Maximum combinational path delay: 3.080ns

==========

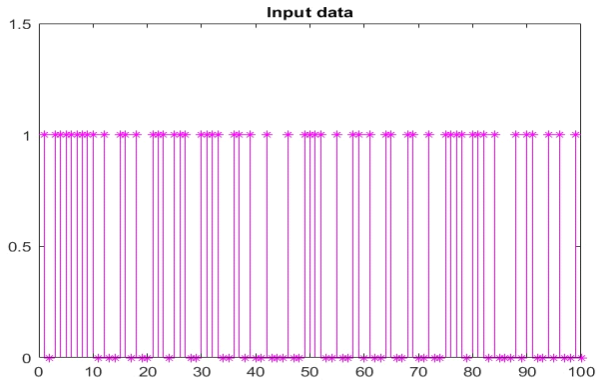
Process "Synthesize - XST" completed successfully

Figure 19: Timing report for the pipeline method

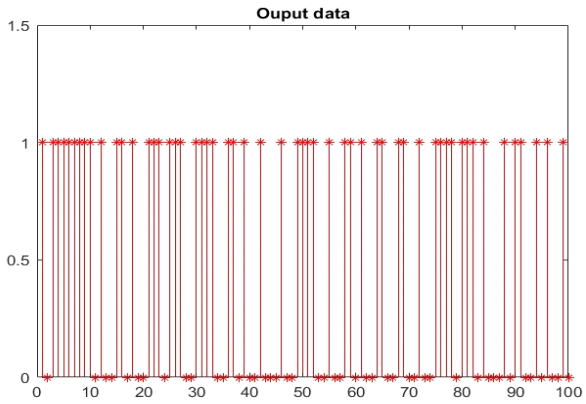
5. Full System Verification

The complete system is verified by comparing the transmitter input to the receiver output with the applied channel effect, it is represented by the AWGN and Rayleigh Channels. The convolutional encoder input was represented by a sequence of binary bits. Fig. 20 (a&b) and Fig. 21 depict the Transmitter's input and the Receiver's output data in MATLAB simulation and VHDL ISim simulator implementation results, respectively. Both simulation and implementation results show that the transmitter's input is the same as the receiver's output. The comprehensive system verification has completed successfully. Fig. 22 depicts

the MATLAB command window output for the same input data stream as VHDL codes. By comparing figures 21 and 22, it is evident that the findings provided by the MATLAB simulation are similar to the ISim simulation results.



(a)



(b)

Figure 20: The complete system MATLAB simulation results (a) The transmitter input (b) The receiver output

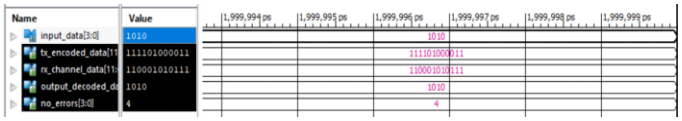


Figure 21: The Full system VHDL ISim simulation results for convolutional coding

Figure 22: A MATLAB command window for whole system convolutional coding

6. Conclusion

This article presented the MATLAB simulation and FPGA implementation of a proposed non-systematic convolutional system that includes a block interleaver and 64-QAM mapping to address AWGN and Rayleigh channel effects. The convolutional code has a rate equal to 1/3, a constraint length of seven, and 63 state transitions. The encoder uses a trellis diagram, while the decoder employs the Viterbi algorithm with hard decision decoding. MATLAB simulation was carried out using version R2023b, and the implementation was done using the Virtex 6 (XC6VLX240T) FPGA kit and VHDL codes from Xilinx package version 14.7, with simulation results presented using the ISim simulator. The system's complexity is evaluated based on FPGA resource usage. A comparison is drawn between FPGA utilization with and without the pipelining algorithm. The non-pipelined system demonstrates satisfactory speed and efficient chip resource utilization, while the pipelining technique balances speed improvement and complexity through the inclusion of additional memory and logic components. The performance of the pipelining approach is dependent on available chip resources and the desired system timing response.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Attia ME, Kabeel AE, Mohamed A, Abdelkader B, and Moataz MA. Optimal Sowmya, K. B., D. N. Rahul Raj, and Sandesh Krishna Shetty. "Error Correction Technique Using Convolution Encoder with Viterbi Decoder." In Sustainable Communication Networks and Application: Proceedings of ICSCN 2020, Springer Singapore, pp. 243-252, 2021. https://doi.org/10.1007/978-981-15-8677-4_20

[2] GodwinPremi, M. S., N. Anusha, S. Kuzhaloli, G. Kumar, and M. Rajmohan. "FPGA implementation of the convolution coding method for industrial automation." In AIP Conference Proceedings, AIP Publishing, vol. 2523, no. 1., 2023. https://doi.org/10.1063/5.0110974

[3] Huleihel, Yara, and Haim H. Permuter. "Low PAPR MIMO-OFDM Design Based on Convolutional Autoencoder." IEEE Transactions on Communications, 2024. https://doi.org/10.48550/arXiv.2301.05017

[4] Shawqi, Farooq Sijal, Lukman Audah, Mustafa Maad Hamdi, Ahmed Talaat Hammoodi, Yassin Salih Fayyad, and Alaa Hamid Mohammed. "An overview of ofdm-uwb 60 ghz system in high order modulation schemes." In 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pp. 1-6. IEEE, 2020.https://doi.org/10.1109/ISMSIT50672.2020.9255175

[5] Prakash, Varshitha, and M. Ramesh Patnaik. "Design and Implementation of Convolution Coding Technique in Industrial Automation." In Frontiers in Intelligent Computing: Theory and Applications: Proceedings of the 7th International Conference on FICTA (2018), Vol. 2, Springer Singapore, 2020, pp. 92-100. https://doi.org/10.1007/978-981-13-9920-6_10

[6] Gómez-Torrecillas, José, Francisco Javier Lobillo, and Gabriel Navarro. "Cyclic distances of idempotent convolutional codes." Journal of Symbolic Computation, 2021, pp. 37-62. https://doi.org/10.1016/j.jsc.2019.10.008

[7] Yaping, S. U. N., D. O. U. Gaoqi, and Y. A. N. Mingliang. "Nested Tail-Biting Convolutional Codes Construction for Short Packet Communications." Radioengineering vol. 30, no. 3, 2021. doi: 10.9781/ijimai.2022.01.004

[8] Mohammed, Hasan Fadhil, and Ghanim A. Al-Rubaye. "BER Performance of Convolutional Coded OFDM System in Different Fading Channels Scenarios with Exact and Simplified LLR." Webology, vol. 19, no. 1, pp. 6300-6321, 2022. https://www.webology.org/abstract.php?id=1347

[9] Zhang, Zhengyu, Dongping Yao, Lei Xiong, Bo Ai, and Shuo Guo. "A convolutional neural network decoder for convolutional codes." In Communications and Networking: 14th EAI International Conference, ChinaCom 2019, Shanghai, China, November 29–December 1, 2019, Proceedings, Part II 14, Springer International Publishing, 2020, pp. 113-125. https://doi.org/10.1007/978-3-030-41117-6_10

[10] Spasov, Dejan. "A Generalization of the Convolutional Codes." 2020. http://hdl.handle.net/20.500.12188/8215

[11] RoyChatterjee, S., K. Sur, and M. Chakraborty. "Study on S-box properties of convolution coder." In Proceedings of International Ethical Hacking Conference 2019: eHaCON 2019, Kolkata, India, Springer Singapore, 2020, pp. 119-128. https://doi.org/10.1007/978-981-15-0361-0_9

[12] Lingxiao Zhao. "Comparisons of PSK, APSK, and QAM over AWGN and fading channels. " ACE, vol. 36 no. 1, pp. 53-63, 2024. DOI: 10.54254/2755-2721/36/20230423.

[13] Katz, Noam. "CommUnet: U-net decoder for convolutional codes in communication." arXiv preprint arXiv:2004.10057, 2020. https://doi.org/10.48550/arXiv.2004.10057

[14] Banerjee, A., Lenz, A., and Wachter-Zeh, A. "Sequential Decoding of Convolutional Codes for Synchronization Errors. " In 2022 IEEE Information Theory Workshop (ITW), IEEE. pp. 630-635, November 2022. https://doi.org/10.48550/arXiv.2201.11935

[15] Zhang, Chen, Guangyu Sun, Zhenman Fang, Peipei Zhou, and Jason Cong. "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks." In Proceedings of the ACM Turing Award Celebration Conference-China 2023, pp. 47-48. https://doi.org/10.1145/3603165.3607390

[16] García Planas, María Isabel, and Laurence Emilie Um. "Convolutional code theory-based steganography technique." International journal of circuits, systems and signal processing vol. 16, pp. 811-821, 2022. https://doi.org/ 10.46300/9106.2022.16.100

[17] Mohammadidoost, Alireza, and Matin Hashemi. "High-throughput and memory-efficient parallel viterbi decoder for convolutional codes on GPU." arXiv preprint arXiv:2011.09337, 2020. https://doi.org/10.48550/arXiv.2011.09337

[18] Venkatesh, C. "Wave pipelined VLSI architecture for a Viterbi decoder using self-reset logic with 0.65 nm technology." International Journal of Applied Science and Engineering vol. 8, no. 1, pp. 65-75, 2021. https://doi.org/10.6703/IJASE.2010.8(1).65

[19] Bhuvanasri, P., T. Geetha Sai Kumari, Sh Raziya, K. Abhishek, and D. Vyshnavi. "An Efficient Low Power Architecture for Vertibe Decoder Using ASIC Application." Journal of Engineering Sciences vol. 14, no. 03, pp. 481-495, 2023. DOI:10.15433.JES.2023.V14I3.43P.54

[20] Elwazan, Aly AE, Abdelhalim AA Zekry, and Hossam LA Zayed. "Matlab Code for LTE Convolutional Code and Viterbi Decoder." International Journal of Engineering Research & Technology (IJERT) vol. 6, no. 3, pp. 578-581, 2017. DOI:10.17577/IJERTV6IS030465

[21] Liao, Jingyi, Kalle Ruttik, Riku Jantti, and Phan-Huy Dinh-Thuy. "Coded Backscattering Communication with LTE Pilots as Ambient Signal." arXiv preprint arXiv:2402.12657, 2024. https://doi.org/10.48550/arXiv.2402.12657