High-Performance Unrolled and Pipelined Architectures for Efficient Encoding Decoding of Polar Codes

[1]Neetu Nayak, [2]Mr. Nagendra Singh Thakur
[1]M.Tech Scholar, Department of ECE, SAM College of Engineering & Technology, Bhoapl, MP, India
[2]Assistant Professor, Department of ECE, SAM College of Engineering & Technology, Bhoapl, MP, India

Abstract- This paper presents a high-performance unrolled and pipelined architecture for efficient encoding and decoding of Polar Codes, targeting next-generation high-throughput and low-latency communication systems. The proposed design exploits full loop unrolling and deep pipelining techniques to maximize parallelism, thereby significantly increasing data throughput while maintaining reliable error-correction performance. By mapping the Polar Code factor graph directly onto dedicated hardware processing elements, the architecture minimizes control overhead and reduces critical path delay. Both encoder and decoder modules are optimized to support scalable code lengths and rates, enabling flexible integration in modern wireless standards such as 5G and beyond. Experimental analysis demonstrates that the proposed architecture achieves superior throughput and energy efficiency compared to conventional folded and semi-parallel designs, making it well suited for real-time, high-speed digital communication applications.

Keywords: *5G, AI, VLSI, Xilinx, Polar Code.*

1. Introduction

The rapid evolution of wireless communication systems and the ever-increasing demand for high data rates, ultra-low latency, and reliable connectivity have driven the development of advanced error control coding techniques for modern standards[1]. In this context, Polar Codes have emerged as a breakthrough channel coding scheme and have been officially adopted in the 5G New Radio (NR) standard for control channel communications. Introduced as the first class of error-correcting codes proven to achieve the Shannon channel capacity, Polar Codes play a crucial role in ensuring robust and efficient data transmission in 5G systems, where reliability and spectral efficiency are of paramount importance[2].

Encoding and decoding of Polar Codes are based on the principle of *channel polarization*, where a set of identical communication channels is transformed into a new set of polarized channels that are either highly reliable or highly unreliable[3]. During the encoding process, information bits are assigned to reliable channels, while unreliable channels are fixed with predefined frozen bits. This structured transformation enables Polar Codes to achieve excellent error-correction performance, especially for short and moderate block lengths, which are commonly used in 5G control channels such as the Physical Downlink Control Channel (PDCCH) and Physical Uplink Control Channel (PUCCH)[4].

In 5G systems, the encoding of Polar Codes must be computationally efficient to support high throughput and low latency requirements. The polar encoder is typically implemented using a recursive construction based on the Kronecker product of a basic kernel matrix, resulting in a highly regular and deterministic structure[5]. This regularity makes Polar Codes well suited for hardware implementation, allowing designers to exploit parallelism, pipelining, and unrolling techniques to meet stringent real-time constraints. Efficient encoding architectures are essential to ensure minimal delay and reduced hardware complexity while supporting multiple code lengths and rates defined by the 5G standard[6].

Decoding of Polar Codes, however, is more challenging and computationally intensive compared to encoding. The most commonly used decoding algorithms include Successive Cancellation (SC), Successive Cancellation List (SCL), and CRC-aided SCL decoding. Among these, CRC-aided SCL decoding has been widely adopted in 5G due to its superior error-correction performance, particularly under low signal-to-noise ratio conditions[7]. Nevertheless, these decoding techniques introduce challenges in terms of latency, memory usage, and power consumption, making efficient decoder design a critical research area for 5G hardware implementations[8].

To address these challenges, advanced architectural techniques such as unrolled, semi-parallel, and fully pipelined decoder designs have been explored. Unrolled architectures eliminate iterative control by mapping the entire decoding process directly onto hardware, enabling extremely high throughput at the cost of increased area[9]. Pipelined architectures further enhance performance by allowing multiple frames to be processed simultaneously, thereby reducing decoding latency and improving resource utilization. Such architectural optimizations are particularly important for 5G base stations and user equipment, where real-time processing and energy efficiency are essential[10].

2. Methodology

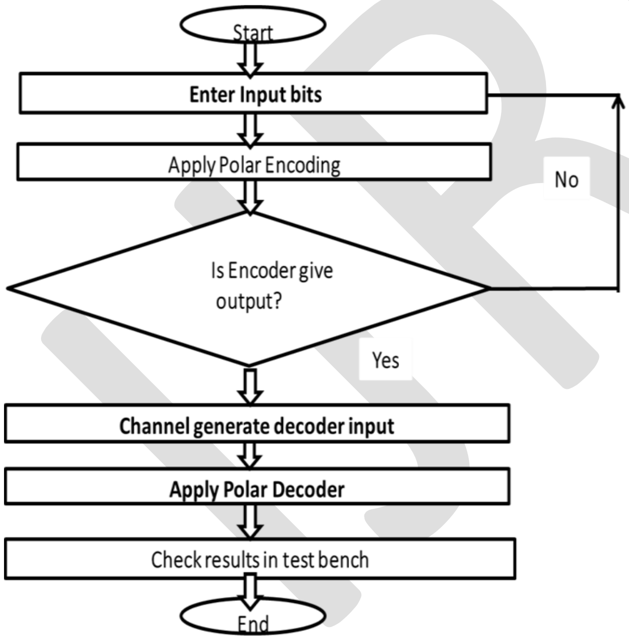


Figure 1: Flow Chart

Step 1: Start and System Initialization The process begins with system initialization, where key polar code parameters such as code length N, number of information bits K, frozen bit locations, decoding method, and channel conditions are defined. This step prepares the encoder, decoder, and test bench environment for operation.

Step 2: Input Bit Generation Information bits are generated and combined with frozen bits. Frozen bits are placed at predetermined unreliable channel positions and are usually set to zero. The complete input vector of length N is formed before encoding.

Step 3: Apply Polar Encoding The polar encoder transforms the input vector using the polar generator matrix constructed from the Kronecker power of the basic kernel matrix. This process spreads information bits across the codeword, achieving channel polarization and generating the encoded output sequence.

Step 4: Encoder Output Verification A decision check is performed to verify whether the encoder has successfully produced the encoded data. If the encoder output is not available, the process waits or repeats the encoding operation. If the output is valid, the encoded bits are forwarded to the channel stage.

Step 5: Channel Modeling The encoded data is transmitted through a simulated communication channel. Noise and interference are added using a channel model such as Additive White Gaussian Noise (AWGN). The channel generates noisy received signals, which serve as input to the decoder.

Step 6: Decoder Input Generation The received noisy signals are converted into suitable decoder inputs, typically in the form of Log-Likelihood Ratios (LLRs). These soft values provide reliability information required for efficient polar decoding.

Step 7: Apply Polar Decoding The polar decoder processes the received LLRs using a decoding algorithm such as Successive Cancellation (SC) or Successive Cancellation List (SCL). Frozen bits are fixed to known values, while information bits are estimated recursively based on likelihood calculations.

Step 8: Output Reconstruction The decoded bit sequence is reconstructed by extracting the estimated information bits from the decoded codeword. This forms the final decoded output.

Step 9: Test Bench Verification The decoded output is compared with the original transmitted information bits in the test bench. Performance metrics such as Bit Error

Rate (BER), Frame Error Rate (FER), and latency are computed to evaluate system accuracy and reliability.

Step 10: End of Process
After verification, the process terminates. The same procedure can be repeated for different signal-to-noise ratios, code lengths, or decoding algorithms to analyze performance under various conditions.

3. Simulation Results

The proposed Polar Code architecture is designed, implemented, and simulated using Xilinx ISE 14.7 design software. Functional verification and result validation are carried out using the ISim simulator, where a dedicated test bench is developed to ensure correct encoding and decoding operations under defined conditions.

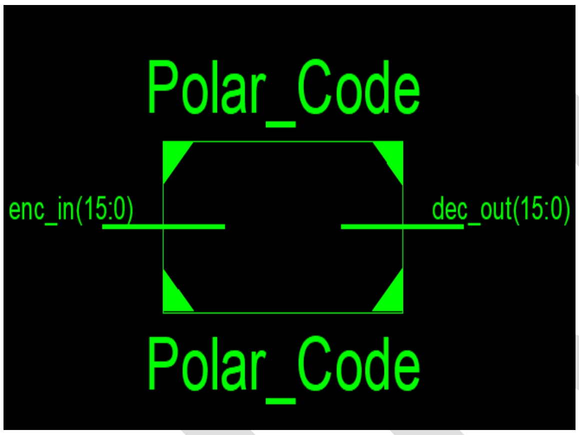


Figure 2: Top view of polar code

Figure 2 illustrates the top-level architecture of the proposed Polar Code system. It consists of three major blocks: the polar encoder, the channel module, and the polar decoder. These blocks collectively perform the complete transmission and reception process.

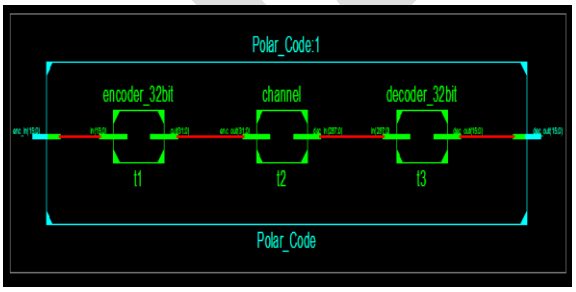
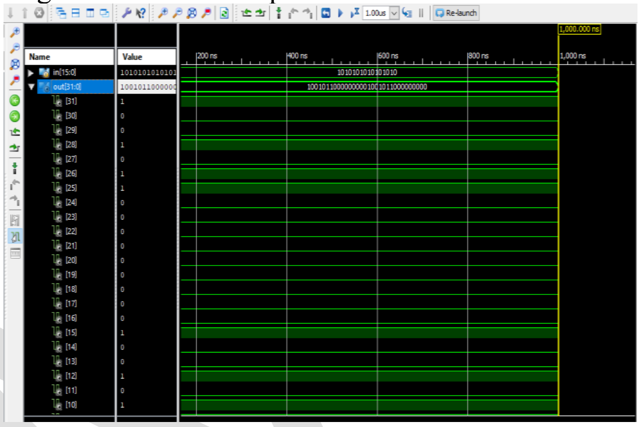


Figure 3: Polar code steps

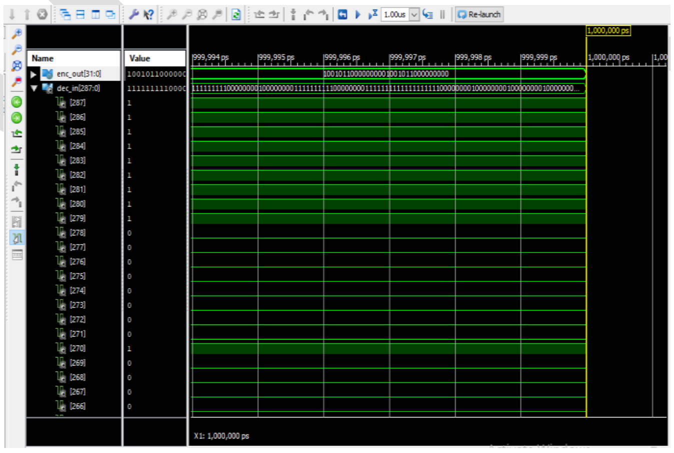
The operational flow of the system is depicted in Figure 3, which explains the major processing stages. In the polar encoder stage, a 16-bit input data sequence is applied and encoded into a 32-bit polar-coded output. This encoded data is then forwarded to the polar channel, where the 32-bit encoder output is expanded and transformed into a 288-bit sequence to simulate channel effects. Finally, the polar decoder processes the 288-bit channel output and successfully recovers the original 16-bit data sequence.



```
Name        Value
in[15:0]    1010101010101
out[31:0]   1001011000000
  [31]      1
  [30]      0
  [29]      0
  [28]      1
  [27]      0
  [26]      1
  [25]      1
  [24]      0
  [23]      0
  [22]      0
  [21]      0
  [20]      0
  [19]      0
  [18]      0
  [17]      0
  [16]      0
  [15]      1
  [14]      0
  [13]      0
  [12]      1
  [11]      0
  [10]      1
```

Figure 4: Test bench encoder output waveform

The functional correctness of the encoder is verified through test bench simulation, as shown in Figure 5. The applied 16-bit input data sequence 1100110011001100 is correctly recovered at the decoder output, confirming error-free transmission and decoding.

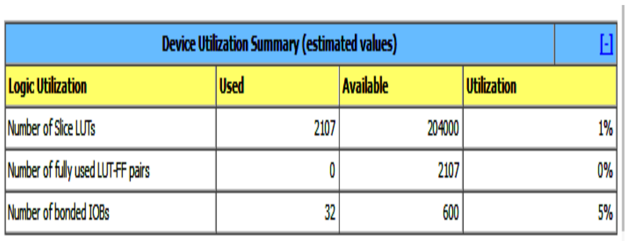


```
Name            Value
enc_out[31:0]   1001011000000
dec_in[287:0]   1111111110000
  [287]         1
  [286]         1
  [285]         1
  [284]         1
  [283]         1
  [282]         1
  [281]         1
  [280]         1
  [279]         1
  [278]         0
  [277]         0
  [276]         0
  [275]         0
  [274]         0
  [273]         0
  [272]         0
  [271]         0
  [270]         1
  [269]         0
  [268]         0
  [267]         0
  [266]         0

X1: 1,000,000 ps
```

Figure 5: Test bench channel output waveform

Figure 5 demonstrates the channel output waveform observed in the test bench. In this case, the 32-bit encoder output 10010110000000001001011000000000 is applied to the channel, which generates a 288-bit decoder input represented in hexadecimal form as hff80403ff00ffffe01008040201008040201ff80403 ff00ffffe01008040201008040201.



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	2107	204000	1%
Number of fully used LUT-FF pairs	0	2107	0%
Number of bonded IOBs	32	600	5%

Figure 6: Device utilization of proposed model

Table 1: Result Comparison

Sr No.	Parameter	Previous Work	Proposed Work
1	Method	Polar decoder	Polar encoder and polar decoder
2	Area	5.35 mm^2	2.33 mm^2
3	Delay	1534ns	139.612ns
4	Power	1072.9 mW	43mW
5	Time	NA	30.48 secs
6	PDP	164153	6003

4. Conclusion

Polar Codes play a vital role in modern 5G communication systems due to their low complexity and strong error-correction capability. Efficient hardware implementation of polar encoding and decoding is essential to meet the high-speed, low-latency, and low-power requirements of next-generation wireless applications. This work focuses on improving performance while reducing resource utilization. An optimized Polar Encoder–Decoder architecture is designed and implemented using FPGA technology. The proposed method integrates both encoding and decoding modules and is validated through simulation using Xilinx ISE tools. Significant improvements are achieved in terms of area, delay, power consumption, and overall efficiency when compared with existing designs. In future work, the proposed architecture can be extended to support higher code lengths and adaptive code rates. Further optimization using advanced decoding algorithms and implementation on newer FPGA or ASIC platforms can enhance performance for beyond-5G and 6G applications.

References

1. Y. Pola, A. Tiwari, S. Sharma and S. S. Chauhan, "Design and Hardware Implementation of Polar Codes Using Verilog for Digital Systems," 2025 IEEE 5th Internatnal Conference on VLSI Systems, Architecture, Technology and Applications (VLSI SATA), Bangalore, India, 2025, pp. 1-6, doi: 10.1109/VLSISATA65374.2025.11070044.
2. S. A. Hebbar, S. K. Ankireddy, H. Kim, S. Oh, and P. Viswanath, Deeppolar: Inventing nonlinear large-kernel polar codes via deep learning, *arXiv preprint arXiv* : 2402.08864, 2024.
3. K. S. Nakul, M. K. C. Reddy, P. Abhiram, and M. Vinodhini, "Row-wise hamming code for memory applications," in *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2024, pp. 30–34. DOI: 10.1109/ICESC60852.2024.10689934.
4. M.-C. Chiu, "Analysis and design of polar-coded modulation," *IEEE Transactions on Communications*, vol. 70, no. 3, pp. 1508–1521, 2022. DOI: 10.1109/TCOMM.2022.3142280.
5. D. Kam, H. Yoo and Y. Lee, "Ultralow-Latency Successive Cancellation Polar Decoding Architecture Using Tree-Level Parallelism," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 6, pp. 1083-1094, June 2021, doi: 10.1109/TVLSI.2021.3068965.
6. C. Ji, Y. Shen, Z. Zhang, X. You and C. Zhang, "Autogeneration of Pipelined Belief Propagation Polar Decoders," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 7, pp. 1703-1716, July 2020, doi: 10.1109/TVLSI.2020.2983975.
7. W. Song, Y. Shen, L. Li, K. Niu and C. Zhang, "A General Construction and Encoder Implementation of Polar Codes," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 7, pp. 1690-1702, July 2020, doi: 10.1109/TVLSI.2020.2983327.
8. R. Kavipriya and M. Maheswari, "High Speed Polar Encoder Architecture For next Generation 5G Applications Using Radix-k Processing Engine," 2019 International Conference on Computer Communication and

Informatics (ICCCI), 2019, pp. 1-4, doi: 10.1109/ICCCI.2019.8821961.

9. R. Shrestha, P. Bansal and S. Srinivasan, "High-Throughput and High-Speed Polar-Decoder VLSI-Architecture for 5G New Radio," 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), 2019, pp. 329-334, doi: 10.1109/VLSID.2019.00075.
10. H. -Y. Yoon and T. -H. Kim, "Generalized Tree Architecture for Efficient Successive-Cancellation Polar Decoding," 2018 IEEE 36th International Conference on Computer Design (ICCD), 2018, pp. 326-333, doi: 10.1109/ICCD.2018.00056.
11. X. Liang, H. Zhou, Z. Zhang, X. You and C. Zhang, "Joint List Polar Decoder with Successive Cancellation and Sphere Decoding," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 1164-1168, doi: 10.1109/ICASSP.2018.8461672.
12. M. Mousavi, Y. Fan, C. Tsui, J. Jin, B. Li and H. Shen, "Efficient Partial-Sum Network Architectures for List Successive-Cancellation Decoding of Polar Codes," in IEEE Transactions on Signal Processing, vol. 66, no. 14, pp. 3848-3858, 15 July15, 2018, doi: 10.1109/TSP.2018.2839586.