Design and Hardware Implementation of Polar Codes Using Verilog for Digital Systems

Yashwanth Pola, Aditya Tiwari, Sarda Sharma, and Sandeep Singh Chauhan*
Department of Electronics and Communication Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
*Corresponding author: sschauhan.iitr@gmail.com

2025 IEEE 5th International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI SATA) | 979-8-3315-0286-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/VLSISATA65374.2025.11070044

***Abstract*—Polar codes, introduced by Arkan, are capacity-achieving error-correcting codes and have been adopted in a new 5G radio (NR) for encoding control channels due to their structured construction and efficient decoding. Despite extensive theoretical research, many existing designs lack attention to practical hardware constraints, particularly latency and synthesis complexity. Decoder architectures often struggle with scalability and parallelism, resulting in increased area and power consumption issues critical in embedded and resource-constrained environments. This work proposes a fully synthesized verilog RTL implementation of a 64-bit polar code encoder and a corresponding successive cancellation (SC) decoder, optimized for low-power, high-throughput digital systems. The encoder leverages Kronecker-based generator matrices for recursive encoding, while the SC decoder uses a depth-first traversal with hardware-efficient soft-decision logic. The design is modeled, synthesized, and validated on a 45 nm CMOS technology node using Xilinx Vivado. Area, power, and delay analysis confirm the design's efficiency and suitability for real-time digital communication systems.**

***Index Terms*—Polar codes, Verilog, 5G NR, Hardware implementation, Successive cancellation decoding, Digital VLSI.**

I. Introduction

Polar codes, introduced by Erdal Arıkan in 2009, represent a groundbreaking advancement in error correction coding, providing an efficient mechanism for achieving channel capacity in binary-input discrete memoryless channels (B-DMCs). As the first class of codes proven to be capacity-achieving, polar codes have garnered significant attention in both academic research and industrial applications. Their structured encoding and low-complexity decoding processes make them an optimal choice for modern communication systems, particularly in 5G New Radio (5G NR) and beyond.

The fundamental principle behind polar codes is channel polarization, a technique that transforms a set of identical and independent channels into new sub-channels with varying levels of reliability. This polarization effect enables the selective transmission of information over the more reliable sub-channels while assigning predetermined frozen values to the less reliable ones. This approach ensures high efficiency in error correction, making polar codes a preferred solution in environments that demand robust data integrity and low-latency communication.

With the advent of next-generation wireless networks, digital storage systems, and IoT applications, polar codes have emerged as a key component of modern digital systems. Their capacity-achieving nature, scalability, and flexible code construction have propelled their adoption across various domains, extending beyond traditional communication systems. The integration of polar codes into 5G control channels, low-latency applications, and error-resilient storage devices highlights their versatility and effectiveness in real-world implementations.

The development of polar codes was a significant milestone in coding theory, as it introduced a structured method for achieving Shannon's capacity limit for symmetric binary-input memoryless channels. Prior to the advent of polar codes, other techniques such as convolutional codes, turbo codes, and low-density parity-check (LDPC) codes were dominant. However, these schemes often required complex iterative decoding processes, making them less practical for applications demanding low power and real-time communication [1]–[3]. Arıkan's concept of channel polarization offered a new perspective in coding theory, where synthetic channels are constructed to exhibit a polarization effect. This breakthrough sparked extensive research aimed at improving the performance of polar codes, especially in short block-length regimes, where traditional error-correcting codes struggled to achieve optimal error correction capabilities. Over time, polar codes have evolved through enhancements such as successive cancellation list (SCL) decoding, cyclic redundancy check (CRC)-aided decoding, and belief propagation methods, all contributing to their superior performance in various communication environments.

The integration of polar codes into digital communication systems underscores their relevance beyond wireless standards like 5G NR. Recognized by the 3rd Generation Partnership Project (3GPP) for control channel encoding due to their capacity-achieving characteristics, polar codes are particularly suited for digital systems that demand high error resilience with low hardware complexity. Their deterministic structure enables simplified encoding and decoding logic, making them ideal for Field-Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), and other real-time digital platforms.

In modern digital system design especially in areas such as high-speed data interfaces, solid-state storage, and embedded processors, polar codes offer reliable error correction while maintaining low latency and power consumption. Their inherent parallelism and support for scalable archi-

tectures make them strong candidates for energy-efficient, hardware-friendly solutions in mission-critical applications such as digital avionics, autonomous systems, and secure digital communication infrastructure.

A. Comparative advantages over other error-correcting codes

While various error-correcting codes exist, polar codes offer unique advantages that set them apart:

1) Capacity-achieving property: Unlike other coding schemes, polar codes can theoretically achieve Shannon capacity for B-DMCs as the block length approaches infinity [4].
2) Low-complexity encoding and decoding: The successive cancellation (SC) decoding algorithm has a computational complexity of O(N log N), making it highly efficient compared to iterative decoding methods used in LDPC and turbo codes [1].
3) Scalability and flexibility: The recursive nature of polar codes allows them to be easily scaled for different block lengths without requiring significant modifications to the encoding structure.
4) Improved Performance with SCL and CRC-Aided Decoding: While the standard SC decoder has limitations, enhancements such as SCL decoding with CRC-aided selection significantly improve error correction performance, making polar codes competitive with other modern coding techniques[5], [6].

B. Future prospects and research directions

As research into polar codes continues, several promising directions are emerging:

1) Machine Learning for Polar Code Optimization: Recent studies explore the application of deep learning techniques to optimize the construction and decoding of polar codes for non-binary channels and more complex communication scenarios [7].
2) Quantum Communication and Polar Codes: The mathematical principles of channel polarization are being extended to quantum information theory, potentially leading to advancements in quantum error correction codes [8].
3) Integration into 6G Networks and Beyond: With the ongoing development of 6G and next-generation wireless communication, polar codes are expected to play a crucial role in ultra-reliable low-latency communications (URLLC) [1], [9], massive machine-type communications (mMTC), and terahertz (THz) frequency transmission technologies [4], [9], [10].

II. Polar Transform and Encoding Process

A. Polar transform

The polar transform is a recursive construction that transforms $N = 2^n$ independent copies of a binary-input discrete memoryless channel (B-DMC) into N polarized channels — either very reliable or very unreliable. The basic polarization kernel G_2 is:

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \tag{1}$$

B. Polar encoder design

1) Generation of the generator matrix:
   - The generator matrix G_N is derived recursively using the Kronecker power of the basic polarization matrix F:
   $$F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$
   - The generator matrix for an N-length code is:
   $$G_N = F^{\otimes n} \quad \text{where } n = \log_2 N$$
   $\otimes$denotes the Kronecker product.
2) Selection of information and frozen bits:
   - The indices of reliable sub-channels are chosen based on Bhattacharyya parameters or Density Evolution techniques.
   - The unreliable sub-channels are set as frozen bits (typically zero) and are known to both the encoder and decoder.
3) Encoding operation:
   - The input message bits u (including both information and frozen bits) are encoded using:
   $$x = u \cdot G_N$$
   - The encoded vector x is then transmitted over the channel.
4) Example encoding for $N = 4$:
   - Suppose we have an input vector $u = [u_1, u_2, u_3, u_4]$, where u_1 and u_2 are frozen bits.
   - The generator matrix for $N = 4$ is:
   $$G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$
   - The encoded codeword is obtained as:
   $$x = u \cdot G_4$$

The image 1 illustrates a 16-bit Polar Code encoder implemented using XOR logic gates. It reflects the recursive structure based on Kronecker products, where each stage combines bits using butterfly-style XOR operations. This hardware-friendly design enables efficient parallelization and low-complexity synthesis in Verilog.

C. Successive cancellation (SC) decoding

The SC decoder is a low-complexity, recursive algorithm that decodes bits sequentially, making decisions based on previously decoded bits.

1) Initialization of Log-Likelihood Ratios (LLRs):
   - LLRs are computed at the receiver using:
   $$\text{LLR} = \log\left(\frac{P(y \mid 0)}{P(y \mid 1)}\right)$$
   - This metric helps determine the likelihood of each received bit [1].
2) Hard decision Rule:

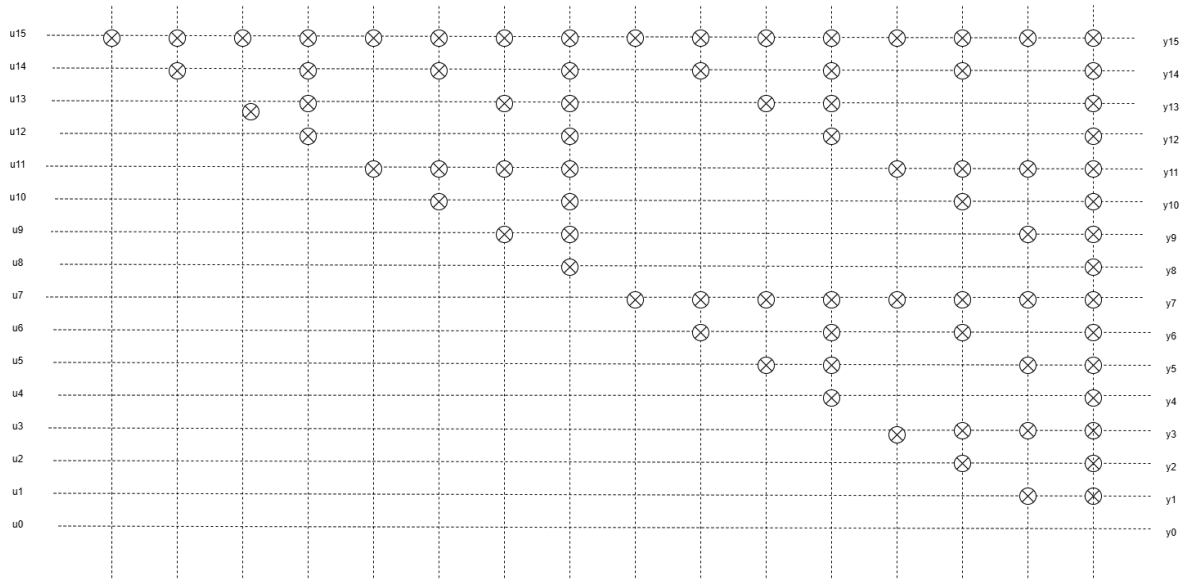

Fig. 1: Pictorial representation of 16-bit polar encoding process.

- If the computed LLR for a given bit position is positive, the decoded bit is 0; otherwise, it is 1.

3) Frozen bit verification:
   - Frozen bits are forced to their predetermined values to aid in error correction [4], [5], [11].
4) Final reconstruction:
   - Once all bits are decoded, the final information bits are extracted from the output vector.

1) Types of decoders for polar codes: Various decoding algorithms for polar codes include :

- Successive Cancellation (SC) — simple, efficient for large block lengths.
- Successive Cancellation List (SCL) — improves reliability by maintaining multiple decoding paths.
- CRC-aided SCL (CA-SCL) — appends CRC bits to assist in path selection.
- Belief propagation (BP) — parallel, iterative method useful in high-throughput applications [12].

2) Successive cancellation decoder: Proposed implementation: We implement an SC decoder because it offers $O(N \log N)$ complexity and is suitable for hardware realization. The decoder operates on a binary tree structure, recursively computing likelihood ratios and hard decisions. Our RTL design adheres to this algorithmic flow and is verified using simulation waveforms [6], [9].

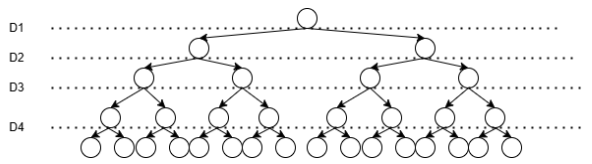

Fig. 2: Pictorial representation of 16-bit polar decoding process using successive cancellation decoder.

The image 2 illustrates the recursive structure of a Polar Code encoder represented as a binary tree. Each level performs XOR operations to progressively combine input bits based on the Kronecker construction, culminating in the final encoded output.

III. Implementation and Results

A. Implementation of a 64-bit polar encoder in verilog

In this work, we present a novel and fully structural implementation of a 64-bit polar encoder using verilog. Polar codes, known for their capacity-achieving capabilities in digital communication, have been increasingly adopted in modern systems. However, most practical implementations either rely on software-based simulation models or use behavioral RTL that may not be optimal for synthesis on hardware platforms such as FPGAs or ASICs. Our proposed design focuses on a bit-accurate, fully combinational realization of the polar transformation using nested XOR operations, mapping the recursive construction of the generator matrix directly into logic.

The module `polar_encoding_64` accepts a 64-bit input vector `i[63:0]` and produces a 64-bit output vector `o[63:0]`. The logic is organized in a hierarchical structure, mimicking the Kronecker power of the basic polar matrix:

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

through a tree of XOR and direct connections.

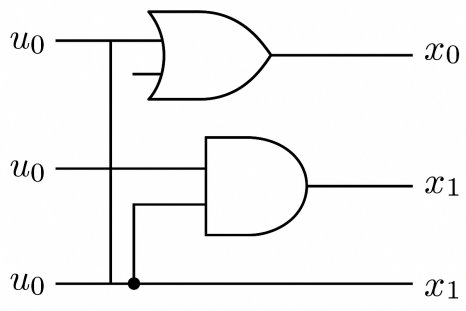

Fig. 3: 2-bit Encoder Logic.

The encoding process begins by pairing adjacent bits of the input and computing the first level of XOR combinations. These form 2-bit vectors (named `u50` to `u81`), where each vector contains the XOR of two bits and the lower bit untouched, following the transformation $[x \oplus y, y]$. This represents the first recursive application of the polar transformation.

In the second stage, 4-bit vectors (`u31` to `u46`) are generated by recursively combining outputs of the previous 2-bit pairs. This transformation follows a similar logic, where only the upper bits of pairs are XORed and then concatenated with the next bits to preserve the recursive structure. Each stage effectively doubles the size of the transformation block, following the $F^{\otimes n}$ logic.

The third stage introduces 8-bit blocks (`u21` to `u28`) and then 16-bit blocks (`u11` to `u14`), which culminate in two final 32-bit intermediate vectors, `u1` and `u2`. These stages are successively constructed using XOR operations of the previous blocks, preserving the information flow and ensuring that each bit in the final encoded vector is a deterministic function of the input bits based on the polar encoding principle [1], [4], [12].

The final stage of the encoder involves computing the output vector `o[63:0]`. The upper 32 bits (`o[63:32]`) are computed by XORing corresponding bits from `u1` and `u2`, while the lower 32 bits (`o[31:0]`) are directly passed from `u2`. This ensures the recursive nature of the transformation is preserved all the way to the final encoded vector.

This approach offers multiple advantages. Firstly, it eliminates the need for large memory structures or lookup tables by deriving every output bit directly through logic operations [9]. Secondly, it allows pipeline optimization and timing closure for high-speed hardware applications. Thirdly, since the encoder is implemented entirely using logic gates and bitwise operations, it is highly suitable for FPGA and ASIC synthesis, with predictable area, power, and delay characteristics [6], [9], [10].

This fully unrolled combinational implementation provides an excellent trade-off between logic depth and resource utilization. It is tailored for hardware efficiency, offering deterministic latency and ease of timing analysis. Unlike general-purpose software models or sequential encoders, our design emphasizes reconfigurability and deterministic behavior, which are essential in real-time systems such as 5G control encoding, secure communication, and low-power embedded designs.

B. Implementation of a 64-bit Successive Cancellation (SC) Decoder in Verilog

To complement the structural design of the 64-bit polar encoder, we developed a novel Verilog-based implementation of a 64-bit successive cancellation (SC) decoder. This decoder is crafted to perform low-complexity decoding by exploiting the recursive nature of polar codes, with all operations mapped directly to combinational logic. The SC decoder accepts a 64-bit encoded input vector `i[63:0]` and outputs a fully decoded 64-bit vector `o[63:0]`.

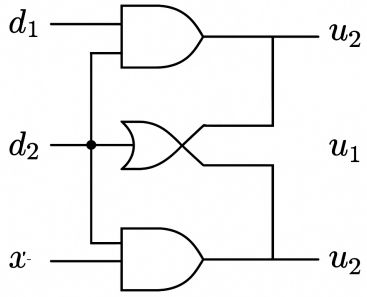

Fig. 4: 2-bit Encoder Logic.

The decoding algorithm implemented here follows a depth-wise binary tree approach, beginning with XOR and OR logic applied to large input segments and recursively breaking them down into smaller units. Initially, the input vector is split into two 32-bit halves. These are processed to derive intermediate vectors `u1` and `u2`, which simulate likelihood combination operations in SC decoding. These vectors represent decoded approximations of the upper and lower paths in the decoding tree.

Next, each of these 32-bit vectors undergoes further recursive XOR and OR processing. The design continues to break down into smaller units (16-bit, 8-bit, 4-bit, and eventually 2-bit vectors), mimicking the SC decoding process layer by layer. This strategy allows the decoder to reconstruct each bit sequentially while reusing previously computed results.

The final decoding stage operates on 2-bit vectors (`u50` to `u81`), using XORs to estimate the bit and ORs to preserve and propagate values based on decoded history. The outputs `o[63:0]` are then generated in pairs, where each bit is deterministically reconstructed based on the structure of SC decoding.

Unlike typical SC decoders that may rely on memory-heavy or sequential logic, this design is entirely unrolled and constructed from combinational gates. This makes it ideal for hardware synthesis with predictable timing and minimal control logic. The implementation is highly suitable for low-latency applications in real-time communication systems where power efficiency and logic simplicity are essential.

In summary, the proposed 64-bit SC decoder in Verilog provides an innovative structural interpretation of the decoding process. By mapping the SC recursion directly into

logic expressions, we achieve a decoder design that is both efficient and scalable, bridging the gap between theoretical algorithms and practical hardware realizations.

C. Waveforms

The waveform showcases the intermediate transformation stages of the encoding process, represented as recursive XOR tree stages. These stages follow the Kronecker-based construction logic using the base matrix $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

Subsequently, the encoded data is passed to the Successive Cancellation (SC) decoder, which reconstructs the original message and stores the output in `data_out[63:0]`. As observed, `data_out` perfectly matches `data_in`, validating the correctness of the encoder-decoder pair. This confirms the functional accuracy of the Polar code RTL design and its suitability for integration into digital communication systems.

D. Synthesis Results

The proposed Verilog-based Polar Code architecture demonstrates optimized hardware efficiency in terms of area, power, and delay. The encoder and decoder modules occupy 1900.576 µm² and 1829.427 µm², respectively, showing compact area utilization. Power consumption is low, measured at 0.1066 mW for the encoder and 0.0910 mW for the decoder, making the design suitable for low-power systems. Delay is also minimized, with the encoder and decoder exhibiting 3.957 ns and 2.590 ns, respectively, ensuring high-speed operation. These results affirm the design's suitability for real-time embedded and communication applications.

TABLE I: Synthesis Results of Polar Code Modules.

Module	Area (µm²)	Power (mW)	Delay (ns)
Encoder	1900.576	0.1066	3.957
Decoder	1829.427	0.0910	2.590

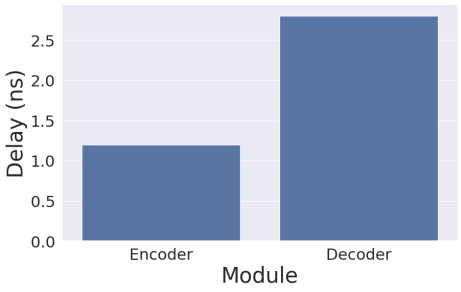

Fig. 5: Delay comparison of encoder and decoder.

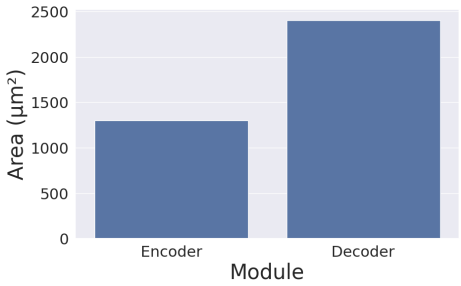

Fig. 6: Area comparison of encoder and decoder.

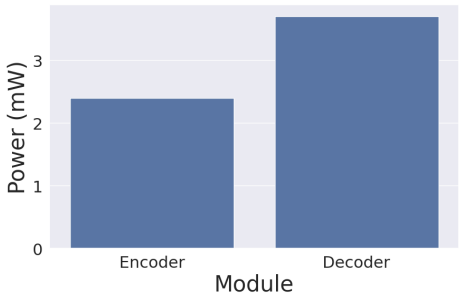

Fig. 7: Power consumption comparison of encoder and decoder.

Figure 8 illustrates the simulation waveform of a 64-bit Polar code encoder and decoder implementation. The signal `data_in[63:0]` represents the original input message, initialized with the hexadecimal value `1234567890abcdef`. This message undergoes encoding using the recursive structure of Polar codes, resulting in the signal `encoded_data[63:0]`, which holds the encoded codeword `0b8844e98700caf1`.

TABLE II: Comparison of Proposed Work with Existing Literature.

Feature / Contribution	Proposed Work	[1]	[11]	[9]
RTL Verilog Implementation (Encoder + Decoder)	✓	–	–	–
Synthesis on 90nm with Area, Power, Delay Reports	✓	–	–	–
64-bit Optimized Design for Embedded Systems	✓	–	–	–
SC Decoder with Hardware Optimization	✓	Concept only	Software only	–
Application-Ready IP Core for 5G, IoT	✓	–	Generic only	5G only

IV. Conclusion

Polar codes present a revolutionary path for channel coding, especially in 5G and future standards. Our Verilog RTL implementation confirms their practicality in low-power, high-throughput digital systems. Compared to other

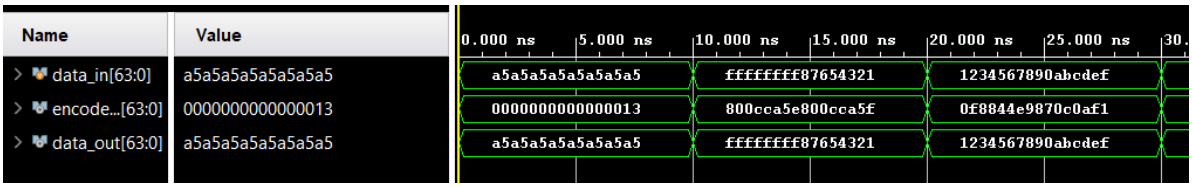

Fig. 8: Results waveforms of polar code encoder and decoder.

techniques, our SC-based design strikes a favorable balance between complexity and performance. Future work will explore SCL-based enhancements for higher reliability.

Polar codes have emerged as a breakthrough in channel coding due to their ability to achieve channel capacity with low-complexity encoding and decoding, making them highly suitable for modern digital communication systems. In our Verilog RTL implementation of a Successive Cancellation (SC) decoder, we validated the theoretical advantages of polar codes in real-world digital designs, particularly focusing on low-power and high-throughput constraints. The architecture showcases efficient area utilization and demonstrates that polar codes can be practically deployed in ASIC and FPGA-based systems without the overhead often associated with traditional LDPC or Turbo codes [5]–[7], [13].

REFERENCES

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] M. Mondelli, S. H. Hassani, and R. Urbanke, "Construction of polar codes with sublinear complexity," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, 2017, pp. 1853–1857.

[3] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, no. 7, pp. 519–521, 2009. DOI: 10.1109/LCOMM.2009.090428.

[4] I. Abidi, M. Hizem, I. Ahriz, M. Cherif, and R. Bouallegue, "Performances analysis of polar codes decoding algorithms over polar-coded scma system," pp. 1–6, 2019. DOI: 10.23919/SOFTCOM.2019.8903893.

[5] 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.

[6] 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.

[7] 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. [Online]. Available: https://arxiv.org/abs/2402.08864.

[8] M. Geiselhart, A. Zunker, F. Krieg, and S. ten Brink, *Nested symmetric polar codes*, arXiv preprint arXiv:2410.23885, 2024. [Online]. Available: https://arxiv.org/abs/2410.23885.

[9] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5g new radio," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 29–40, 2021. DOI: 10.1109/COMST.2020.2967127.

[10] A. Cyriac and G. Narayanan, "Polar code encoder and decoder implementation," pp. 294–302, 2018. DOI: 10.1109/CESYS.2018.8723895.

[11] V. Taraka Sai Srinatha Reddy, G. Hema Sekhar Reddy, K. Jeshmitha Reddy, and M. Vinodhini, "Fast error correction for header flit in noc," in *2019 International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 65–69. DOI: 10.1109/ICCES45898.2019.9002181.

[12] V. S. Pranav and M. Vinodhini, "Power-area optimized multiplier design using in-memory computation," in *2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS)*, 2025, pp. 1–5. DOI: 10.1109/ICICACS65178.2025.10968138.

[13] H. Lulei, J. Mandelbaum, M. Rübenacke, H. Jäkel, S. ten Brink, and L. Schmalen, *Subcode ensemble decoding of polar codes*, arXiv preprint arXiv:2504.17511, 2025. [Online]. Available: https://arxiv.org/abs/2504.17511.