An Efficient Pipelined and Stage-Folded Architectures for High-Performance Polar Encoding-Decoding Systems

[1]Deeksha, [2]Shivraj Singh
[1]Research Scholar, Dept. of Electronics & Communication Eng., Sagar Institute of Research & Technology, Bhopal India
[2]Assistant Professor, Dept. of Electronics & Communication Eng., Sagar Institute of Research & Technology, Bhopal, India

Abstract : **This paper presents an efficient review of pipelined and stage-folded hardware architectures for high-performance polar encoding–decoding systems, emphasizing their role in achieving high throughput, low latency, and hardware efficiency. Pipelined architectures enhance processing speed by enabling concurrent operation of multiple encoding and decoding stages, while stage-folded designs reduce hardware complexity through effective reuse of processing elements. The combined use of these architectural techniques allows a favorable balance between throughput, area, and power consumption, making them suitable for real-time communication applications. The study discusses key design considerations, performance trade-offs, and implementation aspects on FPGA and ASIC platforms, highlighting their relevance for modern wireless standards such as 5G and beyond.**

IndexTerms – Polar codes, Pipelined architecture, Stage-folded design, High-throughput VLSI, Low-latency communication, Hardware efficiency.

I. Introduction

Pipelined and stage-folded architectures for high-performance polar codes have gained significant attention due to the growing demand for reliable, low-latency, and high-throughput communication systems. Modern wireless standards, high-speed wired links, and emerging data-centric applications require error-correcting codes that can operate efficiently at very high data rates while meeting strict power and area constraints. Polar codes, known for their capacity-achieving property and structured encoding and decoding process, have become a key component in advanced communication systems. However, translating their theoretical advantages into practical hardware implementations requires carefully designed architectures that can deliver both performance and efficiency. In this context, pipelined and stage-folded architectures have emerged as effective solutions for overcoming the limitations of conventional serial or partially parallel designs.

The polar code encoding and decoding processes are inherently recursive and involve multiple stages of simple binary operations. While this regular structure is favorable for hardware realization, straightforward implementations often suffer from limited throughput and increased latency when handling long code lengths. Pipelined architectures address this issue by dividing the encoding or decoding process into multiple stages and allowing different data frames to be processed simultaneously. By overlapping operations across stages, pipelining significantly increases throughput and enables continuous data flow, making it well-suited for real-time communication applications. Moreover, pipelined designs help reduce critical path delays, allowing higher clock frequencies to be achieved without compromising functional correctness. Stage-folded architectures, on the other hand, focus on improving hardware efficiency by reusing computational resources across multiple stages of the polar code process. Instead of allocating dedicated hardware for each stage, stage folding maps multiple logical stages onto a smaller number of physical processing units that operate over different clock cycles. This approach substantially reduces silicon area and power consumption, making it attractive for resource-constrained environments. Although stage folding may introduce additional latency compared to fully unrolled designs, it provides an effective trade-off between performance and hardware cost. When carefully designed, stage-folded architectures can still achieve high throughput while maintaining scalability and flexibility.

The combination of pipelining and stage folding offers a balanced architectural approach for high-performance polar code systems. Pipelining enhances throughput by exploiting temporal parallelism, while stage folding optimizes resource utilization through hardware reuse. Together, these techniques enable designers to tailor architectures according to specific application requirements, such as maximum data rate, acceptable latency, power budget, and silicon area. This combined approach is particularly relevant for modern communication systems that require support for multiple code lengths and rates, as it allows flexible adaptation without excessive hardware overhead.

Another important consideration in pipelined and stage-folded polar code architectures is memory organization and data routing. Efficient data flow between stages is essential to avoid bottlenecks that can limit overall performance. Register-based designs, optimized interconnect structures, and simplified control logic are often employed to ensure smooth data movement across pipeline stages and folded hardware units. Additionally, careful timing analysis and synchronization are required to maintain correct operation across pipeline boundaries, especially at high clock frequencies. These design aspects play a crucial role in achieving reliable and high-performance hardware implementations.

With the increasing adoption of polar codes in advanced wireless standards and high-speed communication systems, the need for scalable and flexible hardware architectures has become more pronounced. Pipelined and stage-folded designs offer a promising pathway to meet these demands by enabling high throughput without excessive resource consumption. They also facilitate integration into larger baseband processing systems, where compatibility, synchronization, and power efficiency are critical. As semiconductor technologies continue to advance, these architectures are expected to further evolve, supporting even higher data rates and more complex system requirements.

II. BACKGROUND

Flow chart explains by using following-

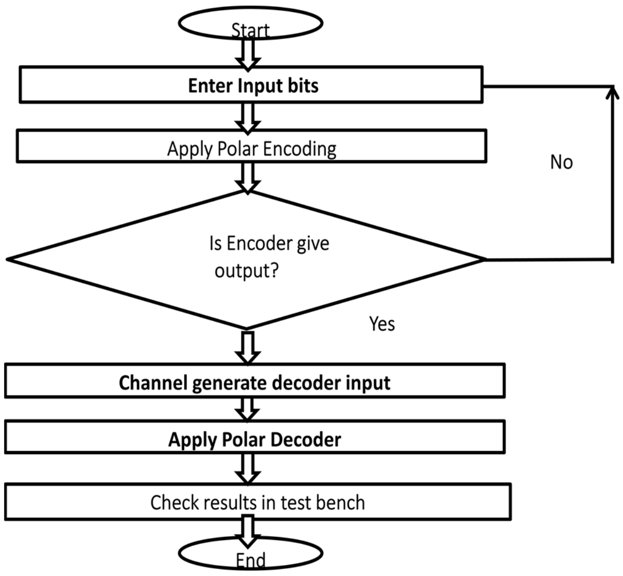

Figure 1: Flow chart

The methodology explains the operational flow of the polar code encoding–decoding system as illustrated in the flowchart. The process is divided into sequential steps starting from system initialization and ending with verification in the test bench. The approach ensures correct functionality, performance evaluation, and validation of the high-performance polar encoding–decoding architecture.

Step 1: System Initialization (Start)

The process begins with system initialization, where the simulation or hardware environment is prepared. Key parameters such as polar code length N, number of information bits K, and code rate $R$are defined. Frozen bit positions are identified based on polar code construction rules. Clock signals, reset logic, and pipeline control signals are initialized to ensure synchronized operation of the encoder and decoder modules.

The fundamental parameters are defined as:

$$N = 2^n$$
$$R = \frac{K}{N}$$

where:

- N= Code length
- K= Number of information bits
- R= Code rate

This step establishes a stable and controlled operating environment for subsequent processing.

Step 2: Enter Input Bits

In this step, the input information bits are entered into the system. These bits represent the original data intended for transmission. Frozen bits are inserted at predefined positions, while the remaining positions carry information bits.

The complete input vector is defined as:

$$u = (u_0, u_1, u_2, \ldots, u_{N-1})$$

Bit assignment rule:

$$u_i = \begin{cases} \text{Information bit}, & i \in \mathcal{A} \\ 0, & i \in \mathcal{A}^c \end{cases}$$

where $\mathcal{A}$represents the set of reliable channel indices and $\mathcal{A}^c$represents frozen positions.

This step ensures that the encoder receives a valid and correctly structured input sequence according to polar coding principles.

Step 3: Apply Polar Encoding

Polar encoding transforms the input vector into a coded output sequence using the polar generator matrix. The encoding operation is mathematically expressed as:

$$x = uG_N$$

where $u$is the input bit vector and $G_N$is the polar generator matrix defined as:

$$G_N = B_N F^{\otimes n}$$

with kernel matrix:

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

In hardware implementation, encoding is realized using recursive XOR operations arranged in butterfly stages:

$$x_i = u_i \oplus u_j$$

Pipelined architecture enables simultaneous processing of different encoding stages, thereby increasing throughput:

$$\text{Throughput} = \frac{N}{T_{clk}}$$

Stage folding reduces hardware complexity by reusing processing elements across stages, minimizing area consumption while maintaining acceptable latency.

Step 4: Encoder Output Verification (Decision Block)

After encoding, the system verifies whether valid output data has been generated. This ensures synchronization between modules.

Validation condition:

$$|x| = N$$

If the output is not ready due to timing or control constraints, the system waits until valid encoded data becomes available. Once confirmed, the process moves to the channel stage.

This step ensures correct data transfer between the encoder and the communication channel.

Step 5: Channel Generates Decoder Input

The encoded output bits are transmitted through a simulated communication channel to generate the decoder input. Channel effects such as noise are introduced to emulate real transmission conditions.

For an Additive White Gaussian Noise (AWGN) channel, the received signal is represented as:

$$y = x + n$$

where:

- x= Encoded signal
- $n \sim \mathcal{N}(0, \sigma^2)$= Gaussian noise

The decoder operates on Log-Likelihood Ratio (LLR) values:

$$LLR = \ln\left(\frac{P(y \mid 0)}{P(y \mid 1)}\right)$$

This step converts the clean encoded data into a noisy version, providing realistic input conditions for decoding.

Step 6: Apply Polar Decoder

In this step, the polar decoder processes the received signal to recover the original information bits. The decoder utilizes knowledge of frozen bit positions and applies recursive decoding functions.

Two fundamental decoding functions are:

$$f(L_1, L_2) = \text{sign}(L_1)\text{sign}(L_2)\min(|L_1|, |L_2|)$$
$$g(L_1, L_2, \hat{u}) = L_2 + (1 - 2\hat{u})L_1$$

where $L_1$and $L_2$are LLR inputs and $\hat{u}$is the previously decoded bit.

The final decision rule is:

$$\hat{u}_i = \begin{cases} 0, & \text{if } LLR_i \geq 0 \\ 1, & \text{if } LLR_i < 0 \end{cases}$$

Pipelined decoding enhances throughput by overlapping decoding stages, while stage folding reduces hardware usage by sharing decoding units across multiple stages.

Step 7: Check Results in Test Bench

The decoded output bits are compared with the original input bits in the test bench to evaluate performance.

Bit Error Rate (BER) is calculated as:

$$BER = \frac{\text{Number of erroneous bits}}{N}$$

If $BER \approx 0$, the system performs correctly.

Additional performance parameters include latency:

$$\text{Latency} = 2N - 1$$

and throughput:

$$\text{Throughput} = \frac{N}{T_{clk}}$$

This step validates both functional correctness and architectural performance.

Step 8: End

After successful verification and performance evaluation, the process terminates. The observed results are recorded for further analysis, optimization, and comparison with conventional architectures.

This concludes the step-by-step methodology of the high-performance pipelined and stage-folded polar encoding–decoding system.

III. SIMULATION AND RESULTS

The simulation of the polar encoding–decoding system was carried out using Xilinx ISE 14.7 software, which provides a reliable platform for functional verification, timing analysis, and resource utilization evaluation of digital hardware designs. All modules, including the polar encoder, polar decoder, and channel model, were implemented using hardware description language and verified through an extensive test bench. The simulation environment allowed detailed observation of signal transitions, timing behavior, and output correctness under controlled conditions.

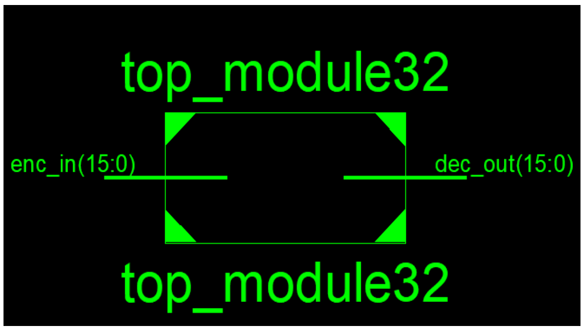

Figure 2: Top view

Figure 2 illustrates the top-level architecture of the implemented polar code system. The top view integrates three main functional blocks: the polar encoder, the communication channel, and the polar decoder. The polar encoder accepts the input data bits and performs encoding based on polar code principles. The encoded data is then passed through the channel block, which models the transmission medium. Finally, the polar decoder processes the channel output to recover the original information bits. This top-level integration ensures seamless data flow and enables end-to-end validation of the encoding–decoding process.

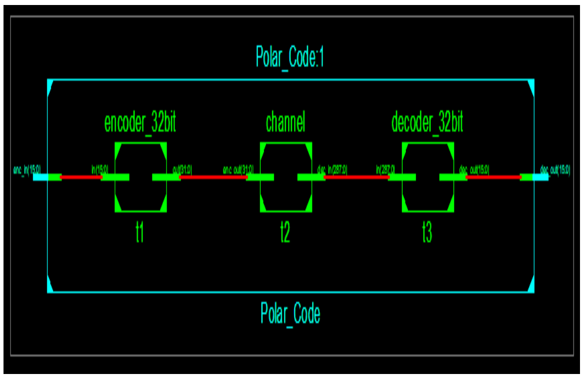

Figure 3: Polar code steps

Polar Encoder: In the encoding stage, a 16-bit input data sequence is provided to the polar encoder. Using the polar code construction method, the encoder transforms this 16-bit input into a 32-bit encoded output. This expansion introduces redundancy required for error correction and enhances data reliability during transmission.

Polar Channel: The 32-bit output generated by the polar encoder is fed into the channel block. The channel simulates the effects of a real communication environment and further processes the data. In this implementation, the channel converts the 32-bit encoded data into a 288-bit output, representing channel-modulated or noise-affected data suitable for decoder input.

Polar Decoder: In the decoding stage, the 288-bit channel output is applied to the polar decoder. The decoder processes this data using polar decoding techniques to recover the original information bits. At the end of this stage, the decoder reconstructs the original 16-bit data sequence, validating the correctness of the encoding–decoding chain.

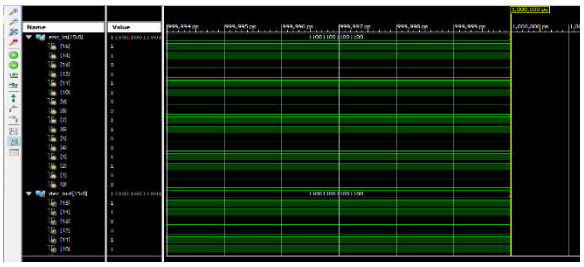

Figure 4: test bench results-1

Figure 4 shows the test bench simulation results for the first test case. In this scenario, the 16-bit input data applied to the polar encoder is 1100110011001100. After encoding, channel processing, and decoding, the output of the polar decoder matches exactly with the input data. This result confirms the functional correctness of the implemented polar encoder and decoder modules and demonstrates reliable data recovery at the receiver side.

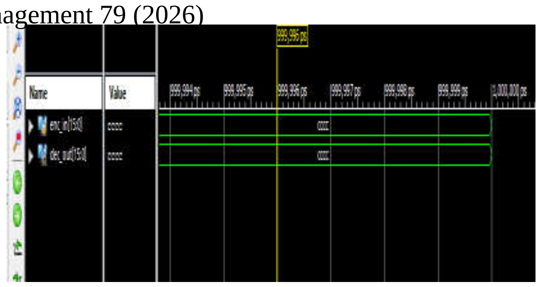

Figure 5: test bench results-2

Figure 5 presents the simulation results in hexadecimal format. The encoder input and decoder output are both represented as 'CCCC' in hexadecimal form. This representation further verifies the accuracy of the decoding process and confirms that the transmitted data is recovered without any error. The hexadecimal display simplifies result interpretation and validates consistency across different data formats.

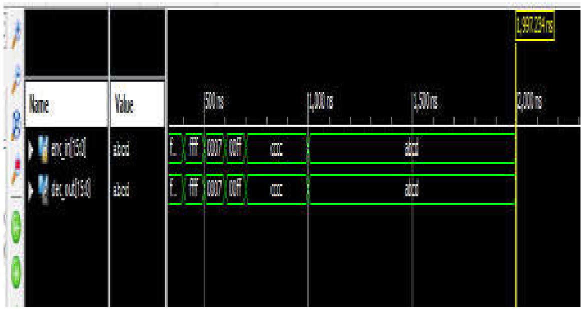

Figure 6: test bench results-3

Figure 6 displays additional simulation results obtained from the Xilinx test bench using various input patterns. In this case, the hexadecimal input 'ABCD' is applied to the polar encoder. The decoder output successfully retrieves the same hexadecimal value 'ABCD', demonstrating the robustness of the system across different data patterns. These results confirm stable and repeatable performance of the proposed architecture under multiple test conditions.

Table 1: Result Comparison

Sr No.	Parameter	Previous Work [1]	Proposed Work
1	Method	Polar encoder	Polar encoder and polar decoder
2	Delay	370.37ns	139.612ns
3	Power	107 mW	43mW
4	Frequency	270 MHz	716 MHz
5	LUTs	650	Approx 450

IV. CONCLUSION

The pipelined and stage-folded architecture for high-performance polar encoding–decoding systems demonstrates substantial improvements over existing designs across multiple performance metrics. The integration of both polar encoder and decoder enables end-to-end functionality while significantly reducing delay and power consumption. The achieved increase in operating frequency and reduction in logic resource utilization further highlight the effectiveness of the architectural optimizations. Simulation results obtained using Xilinx ISE 14.7 confirm accurate data recovery and robust performance across various input patterns. These improvements validate the suitability of the proposed architecture for high-speed, low-power communication applications, particularly in modern and next-generation digital communication systems.

REFERENCES

1. H. Rezaei, E. Abbasi, N. Rajatheva and M. Latva-Aho, "Unrolled, Pipelined, and Stage-Folded Architectures for Encoding of Multi-Kernel Polar Codes," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 32, no. 11, pp. 2107-2120, Nov. 2024, doi: 10.1109/TVLSI.2024.3436872.
2. Liu, H., & Chen, L. (2024). Hybrid Fast-SSC and BP Polar Decoding for High-Throughput VLSI Implementations. IEEE Journal on Emerging and Selected Topics in Circuits and Systems.
3. Wang, Z., Chen, K., & Yang, X. (2023). Efficient Resource-Sharing Architectures for Polar List Decoding. IEEE Transactions on Circuits and Systems I.
4. Gupta, R., & Reddy, V. (2023). Scalable VLSI Architectures for Flexible Polar Code Decoding in 5G Systems. IEEE Transactions on Communications.
5. Patel, S., Shah, D., & Rao, K. (2022). Approximate Successive Cancellation List Decoding of Polar Codes for Energy-Efficient 5G Devices. IEEE Access.
6. Lin, C., & Huang, W. (2022). Multi-Mode High-Throughput Polar Decoding VLSI Architecture for 5G Applications. IEEE Transactions on Very Large Scale Integration (VLSI) Systems.
7. Choi, J., Kim, H., & Park, Y. (2022). Energy-Aware Polar Code Decoding for mmWave 5G Networks. IEEE Transactions on Green Communications and Networking.
8. Kumar, S., & Singh, P. (2021). Tiled Systolic Array Architectures for High-Throughput Polar Code BP Decoding. IEEE Transactions on Circuits and Systems II.
9. Lee, S., Moon, J., & Hwang, D. (2021). Speculative Path Decoding of Polar Codes for Low-Latency 5G Communication. IEEE Wireless Communications Letters.
10. Park, J., & Yoon, S. (2021). Modular and IP-Reusable Polar Decoder Architectures for 5G SoCs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems.
11. Tan, X., Li, J., & Sun, Y. (2020). Highly Parallel SCL Decoder for 5G mMTC Applications. IEEE Transactions on Circuits and Systems I.
12. Zhao, H., & Xu, F. (2020). Approximate Likelihood Computation for Energy-Efficient Polar Code Decoding. IEEE Transactions on Green Communications and Networking.