Parallel encoding and decoding algorithms with optimized hardware architecture for polar codes to reduce complexity and processing time

Dang Trung Hieu[1], Tran Van Nghia[2*], Nguyen Thi Thuy[1]

[1]Electric Power University, 235 Hoang Quoc Viet, Bac Tu Liem, Hanoi, Vietnam;
[2]Air Force - Air Defense Technical Institute, Khuong Trung, Thanh Xuan, Hanoi, Vietnam.
[*]Corresponding author: nghiamosmipt@gmail.com
Received 08 Dec. 2024; Revised 12 Mar. 2025; Accepted 04 Apr. 2025; Published 15 Apr. 2025.
DOI: https://doi.org/10.54939/1859-1043.j.mst.102.2025.51-59

ABSTRACT

This paper presents the development of a simplified equivalent algorithm for fully parallel encoding and decoding of polar codes, which reduces computational complexity and minimizes processing latency compared to existing parallel encoding and decoding methods. The algorithm is implemented in the System Generator/Vitis Model Composer with parameterization capabilities, enabling easy maintenance and adaptation of FPGA designs for future standards. The design accuracy is validated through MATLAB simulations of the standard 3GPP code, and RTL synthesis using Vivado significantly improves latency and throughput.

Keywords: Polar codes; Parallel encoding and decoding; Optimized hardware architecture.

1. INTRODUCTION

Polar coding is a Forward Error Correction (FEC) method designed to achieve channel capacity, even in high Signal-to-Noise Ratio (SNR) environments. Its scalability and low complexity make it suitable for various applications. Consequently, the 3rd Generation Partnership Project (3GPP) has selected polar coding for the 5G communication standards [1]. However, the next-generation radio standards impose stringent throughput and latency requirements, necessitating that Polar encoders and decoders not only to deliver high error correction performance but also to satisfy these criteria under conditions of large-scale data.

Some encoding and decoding algorithms have been proposed to enhance the performance of polar codes, including Successive Cancellation (SC)-based encoding [2-4], List Decoding [5], and variants such as Shifted-Pruning [6], Segmented Flipped SC List Decoding [7], and Multi-bit Decision SC List Decoding [8]. High-speed encoder architectures, such as partially parallel encoders [9, 10] and FPGA-optimized designs for reduced complexity [11] have also been developed. However, increasing code-word length, especially in real-time broadband wireless networks, these architectures face challenges related to higher complexity and processing delays [12-14]. To address this, specialized decoder architectures leveraging Simplified Successive Cancellation (SSC) based techniques and FPGA optimization have been proposed to achieve high-throughput polar decoding [15].

The SC algorithm was the first decoding method introduced for polar codes. However, in comparison with LDPC decoders, it has limitations. First, the SC algorithm has low throughput due to the sequential nature of SC decoding, limiting parallelism and processing speed [2, 4]. Second, polar codes only achieve asymptotic capacity and have worse error correction performance than LDPC codes at moderate lengths [10]. These issues were addressed by the SCL algorithm, which enables parallel processing of the L most probable decoding paths and incorporates a short cyclic redundancy check (CRC) code to identify the correct code-word among the L estimates [5, 6]. Additionally, the log-likelihood ratio (LLR) update functions have been optimized to minimize data copying, and several simplifications based on identifying special nodes have been developed to reduce computational complexity [8, 13].

This paper proposes a low-complexity, fully parallel encoding and decoding algorithm for polar codes to achieve high processing speed and low latency while meeting 5G 3GPP standards. The hardware design of the proposed algorithm based on FPGA has also been implemented to ensure efficient logic resource utilization.

2. DESCRIPTION OF PROPOSED ALGORITHM AND HARDWARE ARCHITECTURE

2.1. The proposed parallel encoding algorithm and its hardware architecture

A fully parallel encoding architecture was proposed in [10] by Arikan to increase the coding speed and reduce processing latency. In this architecture, the entire encoding process was implemented simultaneously in a clock period. It utilizes a generator matrix, G_N, and applies XOR operations to input message bits *x* to produce polar code *y*. The generator matrix is derived through the application of the Kronecker power operation on the kernel matrix, *K*, as follows:

$$G_N = K^{\otimes n} \tag{1}$$

$$y = x.G_N \tag{2}$$

where $K = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $n = \log_2 N$, and *N* denotes the length of the polar code.

The parallel encoder requires n stages to generate a polar code of length *N*. The first of these operations combines two independent copies of the base channel *Z* (a binary discrete memory less channel, B-DMC) to obtain the channel Z_2. At each subsequent stage, two independent copies of the channel from the previous stage, $Z_{N/2}$, are combined to produce the channel Z_N [10].

Define, R_N as a permutation operation that perform a reverse shuffle, Z_N is channel combining: Z_N: $X^N \rightarrow Y^N$ with transition probabilities [10]:

$$Z_N\left(y^N \middle| x^N\right) = Z_{N/2}\left(y_1^{N/2} \middle| x_1^{N/2} \oplus x_{N/2+1}^N\right) \cdot Z_{N/2}\left(y_{N/2+1}^N \middle| x_{N/2+1}^N\right) \tag{3}$$

This presents a significant challenge for its adoption in broadband wireless systems such as 5G and future generations, which require a long polar code to achieve high error-correcting performance. Therefore, further research is required to improve the parallel polar encoder, making it more suitable for broadband wireless systems.

The following algebraic transformation forms the mathematical foundation for constructing the hardware structure of the proposed algorithm:

Suppose $N = 2^n$ with $n \geq 0$, I_m is *m*-dimensional identity matrix, with any $m \geq 1$. The generator matrix G_N is constructed according to recursive formula as follows [10]:

$$G_N = \left(I_{N/2} \otimes K\right) R_N \left(I_2 \otimes G_{N/2}\right) \tag{4}$$

Based on algebraic proof, we obtain $\left(I_{N/2} \otimes K\right) R_N = R_N \left(K \otimes I_{N/2}\right)$. As a result, an alternative form of the recursive formula (4) is:

$$G_N = R_N \left(K \otimes I_{N/2}\right)\left(I_2 \otimes G_{N/2}\right) = R_N \left(K \otimes G_{N/2}\right) \tag{5}$$

Substitute $G_{N/2} = R_{N/2}\left(K \otimes G_{N/4}\right)$ into (5) to obtain:

$$G_N = R_N \left(K \otimes \left(R_{N/2}\left(K \otimes G_{N/4}\right)\right)\right) = R_N \left(I_2 \otimes R_{N/2}\right)\left(K^{\otimes 2} \otimes G_{N/4}\right) \tag{6}$$

Repeating this process we obtain:

$$G_N = P_N K^{\otimes n} \tag{7}$$

where $P_N \triangleq R_N\left(I_2 \otimes R_{N/2}\right)\left(I_4 \otimes R_{N/4}\right)\ldots\left(I_{N/2} \otimes R_2\right)$ is a bit-reversal permutation matrix, represented by the following formula:

$$P_N = R_N\left(I_2 \otimes P_{N/2}\right) \tag{8}$$

From the mathematical transformation, instead of performing the bussed-XOR operation before the permutation, the proposed algorithm reverses the execution process (see figure 1).

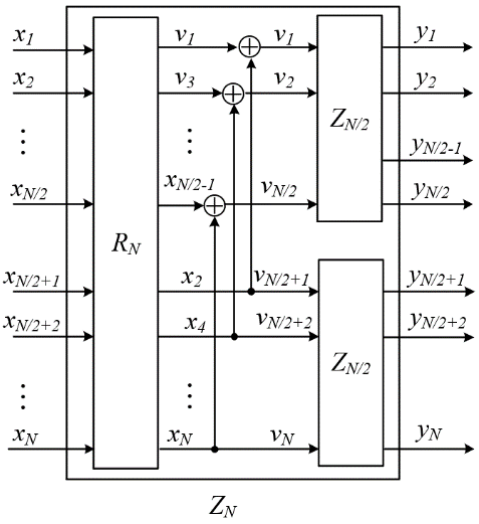

Figure 1. *Proposed recursive architecture for a fully parallel encoder for Polar codes.*

In hardware implementation, a bussed-XOR operator for two *m*-bit inputs is a bitwise operator that performs an XOR operation on each corresponding pair of bits from the two m-bit inputs, producing a new output of the same size, *m* bits. Recursive algorithm 1 describes how the Polar code implementation operates on hardware. In this algorithm, the input data block consists of bits indexed from 1 to *N*. Algorithm 1 is invoked for each input vector *X*. The pipeline architecture's registers are temporarily disregarded to simplify the algorithm description. In the algorithm 1, if bit indexing is performed from 0 to *N-1*, we only need to replace vector **C** with vector **B** in step 5.

Algorithm 1

1. **Initialization:** Determine the length of the code - word *N*; calculate the number of recursive blocks, $M = \log_2 N$; and set the initial input vector $E^{(0)} = X$;

2. **Set** $k = 1$;

3. **Vector Slicing:** Divide the input vector $E^{(k-1)}$ into two equal-sized sub-vectors: $B^{(k)}$ consisting of $N/2$ odd bits and $C^{(k)}$ consisting of $N/2$ even bits. Here, the low- and high-significant bits of $E^{(k-1)}$ become the low-and high-significant bits of $B^{(k)}$ and $C^{(k)}$, respectively.

4. **Bussed-XOR Calculation:** Implement the bussed-XOR operation to create the vector $D^{(k)} = (B^{(k)} \oplus C^{(k)})$.

5. **Construct** $E^{(k)}$: An *N*-bit vector $E^{(k)}$ is obtained by concatenating vectors $C^{(k)}$ and $D^{(k)}$. The bits of $C^{(k)}$ become *N*/2 bits MSB of $E^{(k)}$, while the bits of $D^{(k)}$ are *N*/2 bits LSB of $E^{(k)}$.

6. **Recursive Check:** If $k = M$, set the vector $E^{(k)}$ as the output code-word Y and then proceed to step 7. Otherwise, increment k by 1, and return to step **3**.

7. **End**

The proposed optimal hardware architecture for encoder is shown in figure 2. The pipeline architecture is utilized to achieve high throughput. In this case, the bussed-XOR operation is implemented with a delay of one clock cycle, and the vector **C** in step 5 is also delayed by one clock cycle before forming the vector **E**. The delay of vector **C** is achieved by adding an additional register in the FPGA resource.

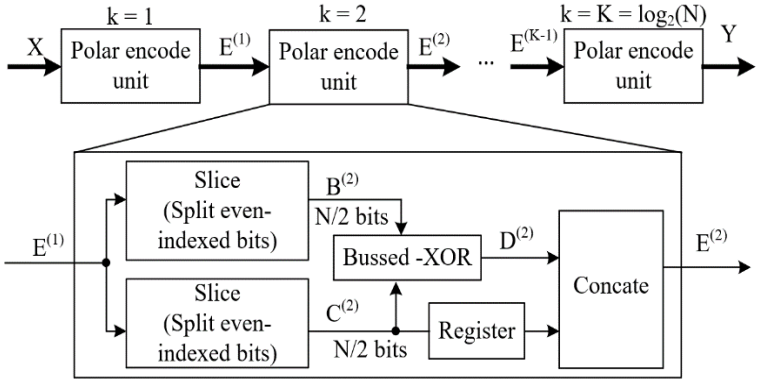

Figure 2. *The proposed optimal hardware architecture for encoder.*

2.2. The proposed parallel decoding algorithm and its hardware architecture

Although computationally simple, the successive cancellation decoder algorithm's execution speed is limited by its inherent serial structure. Decoding x_i needs that all preceding bits $\{x_0, x_1, ..., x_{i-1}\}$ be decoded beforehand, with only one bit estimated at any given time. As a result, various enhancements have been proposed to increase the algorithm's speed by enabling the simultaneous estimation of multiple bits.

The Belief Propagation (BP) algorithm has previously been used to perform inference on graph models, such as Bayesian networks. The BP algorithm, in particular, has demonstrated outstanding success when employed to decode LDPC codes. As a result, highly optimized BP decoder implementations in both hardware and software have been developed over time. The main advantage of a BP decoder lies in its increased parallelism based on the flooding schedule, which differs from the serial scheduling observed in SC decoding. Therefore, employing BP decoding for polar codes will bring huge efficiency gains, avoiding the need to develop complex new hardware designs specifically for polar codes.

Various methods have been proposed for applying the BP algorithm to polar codes. Current multi-bit parallel decoders suffer huge hardware complexity, which increases with code length. This paper presents a fully parallel architecture for a BP-based polar decoder, outlined in algorithm 2. The proposed algorithm reverses the encoding steps described in algorithm 1 on the received data block $\hat{Y}$ to produce the decoded output $\hat{X}$. The data block's bit index ranges from 1 to *N*, corresponding to the LSB and MSB positions.

Algorithm 2

1. **Initialization:** Determine the length of the code-word, *N*; calculate the number of recursive blocks, $M = \log_2 N$; and set the initial input vector $Z^{(0)} = \hat{Y}$;

2. **Set** $k = 1$;

3. **Vector Slicing:** Divide the input vector $Z^{(k-1)}$ into two equal-sized sub-vectors: $J^{(k)}$ consisting of *N*/2 LSB bits and $H^{(k)}$ consisting of *N*/2 MSB bits. Here, the LSBs of $Z^{(k)}$ map to $J^{(k)}$, while

the MSBs correspond to $H^{(k)}$;

4. **Bussed-OR Calculation:** Perform the bussed-XOR operation to generate the vector $U^{(k)} = (J^{(k)} \oplus H^{(k)})$.

5. **Construct** $Z^{(k)}$: An *N*-bit vector $Z^{(k)}$ is obtained by concatenating vectors $U^{(k)}$ and $H^{(k)}$. The low- and high-significant bits of $U^{(k)}$ and $H^{(k)}$ are placed in the low- and high-significant bit positions of $Z^{(k)}$; respectively, with the bits of $U^{(k)}$ mapped to odd bit positions and the bits of $H^{(k)}$ to even bit positions of $Z^{(k)}$.

6. **Recursive Check:** If $k = M$, set the vector $Z^{(k)}$ as the decoded code-word $\hat{U}$ and then proceed to step 7. Otherwise, increment *k* by 1, and return to step 3.

7. **End**

The proposed optimal hardware architecture for decoder is shown in Fig. 3.

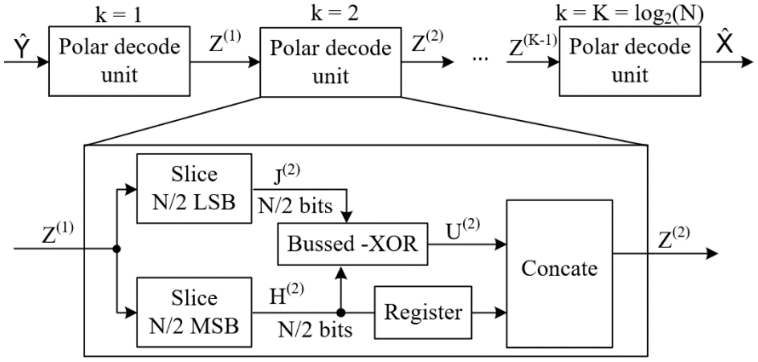

Figure 3. *The proposed optimal hardware architecture for decoder.*

2.3. Analyis of the complexity of proposed algorithms

The complexity of the encoder/decoder can be analyzed based on figure 2 and figure 3. The encoder/decoder contains $log_2 N$ units, with each unit including a register, a 2-input, *N*/2-bit *bussed-XOR* block, a *slice* block, and a *concate* block. The register helps to synchronize the one-clock delay introduced by the *bussed-XOR*. To simplify analysis and comparison, this register is omitted. The *concate* and *slice* blocks, on the other hand, neither require any FPGA resources nor introduce any additional processing delay [16]. As a result, they can be discarded in the complexity analysis and related comparisons. Whereas the *bussed-XOR* block requires *N*/2 XOR operations. Therefore, the complexity of the encoder/decoder is $(N/2)log_2 N$.

The complexity of exiting polar encoder is $O(N)$ for B_N and $O(N \log_2 N)$ for $F^{\otimes n}$ [10]. The complexity reduction ratio of the proposed polar encoder (in %) is:

$$\frac{(N + N\log_2 N) - (N/2)\log_2 N}{N + N\log_2 N} \cdot 100\% = \frac{N + (N/2)\log_2 N}{N + N\log_2 N} \cdot 100\% \qquad (9)$$

For a 5G system with $N = 1024$, this ratio reaches 54.55%

Thus, the proposed decoder's complexity is equivalent to that of the decoder using the original BP algorithm. However, since the delay of each processing block is only one clock cycle, the total delay is at least $\log_2 N$ clock cycles. In addition, the hardware implementation of the proposed decoder is also simpler.

In the parallel SCL decoder, for each information bit, the decoder generates two candidates

for the two possibilities of the bit: one that agrees and one that disagrees with the threshold detection. As a result, the number of candidates doubles, becoming 2L. Although the idea is simple, the implementation is extremely complex in terms of time and memory, since information about each node of each SC encoder needs to be stored, exchanged, and compared as necessary. Specifically, the time complexity is $O(LN^2)$, and the memory capacity is $O(LN^2)$.

This study optimizes the process implementation of the proposed BP algorithm by employing shared storage for soft bits instead of storing all information for each node across all SC encoders. Soft bits are pushed and popped as needed, ensuring that only essential data is retained in memory. This approach reduces time, memory, and computational complexity to $O(LN\log_2(N))$. Consequently, the proposed BP algorithm achieves a processing latency reduced by LN times to $\log_2(N)$.

3. REVIEW AND DISCUSSION OF THE DESIGN

As specified in the 3GPP Standard for 5G wireless systems and illustrated in figure 4, each Polar-coded bit block consists of information, CRC, parity check (PC), and frozen bits, which are interleaved, punctured, and sorted through multiple operations. Normally, the interleaver is only activated in the downlink, while the parity check bits exist only in the uplink. In FPGA designs and testing, to minimize unnecessary complexity, implementations for the downlink and uplink are often separated but still share the same architecture. The simulation results and hardware design implementations presented in this section use the parameters (N, P, Q) = (1024, 512, 8) as specified in the 3GPP Standard for 5G wireless systems, where N is the Polar code-word output length, P is the number of input information bits, and Q is the number of CRC code bits.

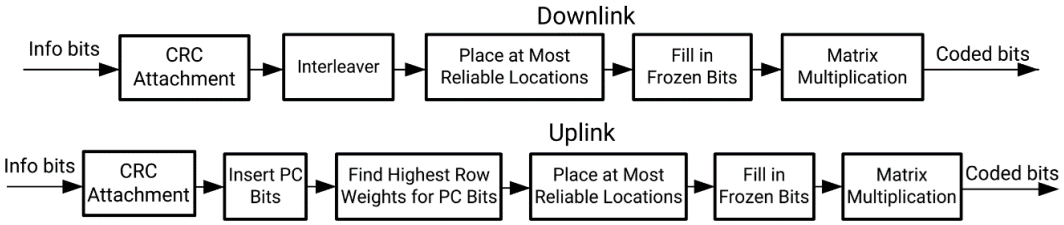

Figure 4. *Block diagram of data transmission processing for 5G uplink and downlink.*

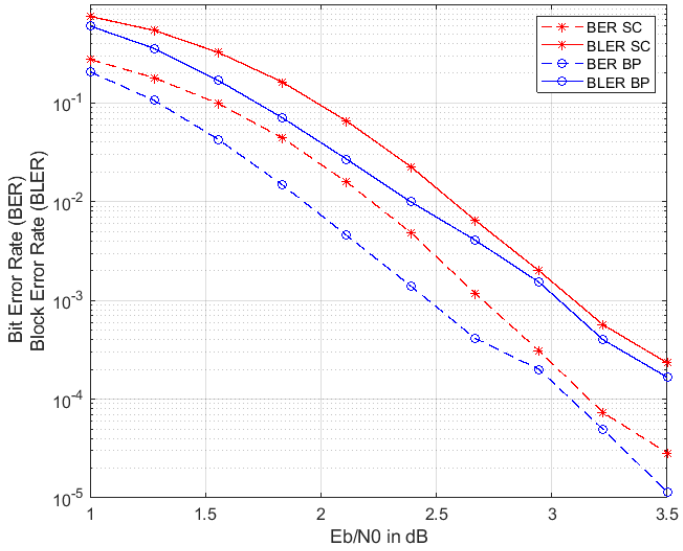

Figure 5. *Comparison of BER and BLER performance between BP and SC decoding.*

Figure 5 shows a comparison of the error performance of BP decoding versus SC decoding

for Polar codes. The simulations are performed on BPSK-modulated data over an AWGN channel. It can be observed that the BP decoder outperforms the SC decoder.

The 3GPP standard for 5G wireless systems recommends using an encoder/decoder with additional parity bits added to the coding vector to enhance BER performance. Two min-sum LDPC decoders operate alternately to decode the received vector, providing an updated estimate of the transmitted code-word. The first decoder processes the polar code, while the second decoder deals with the parity bits. Figure 6 shows that the BP decoder with additional parity bits can improve BER performance by about 0.3 dB.

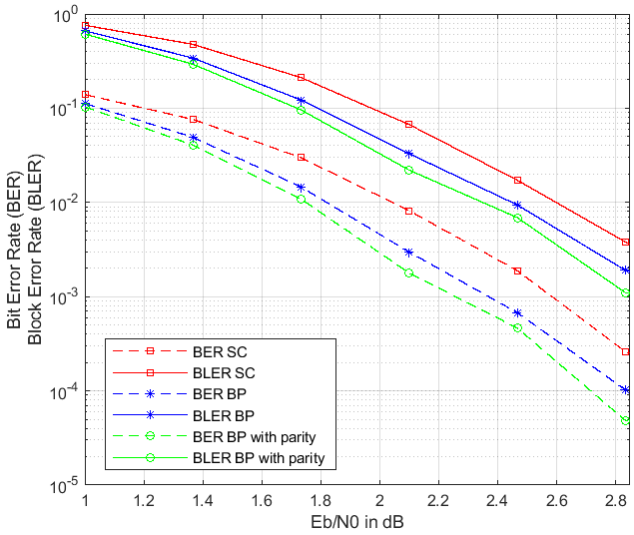

Figure 6. *Improvement in BER performance of the BP decoder with additional parity bits.*

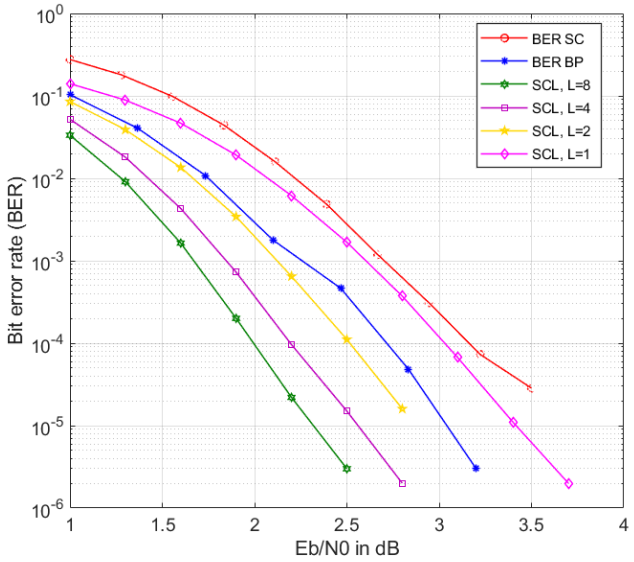

Figure 7. *Comparison of BER performance between the encoder/decoder using the proposed BP algorithm and those utilizing other popular algorithms.*

Figure 7 compares the BER performance of the encoder/decoder using the proposed BP algorithm with those using other popular algorithms. The simulation results show that, compared with the SC algorithm, the proposed BP algorithm achieves significantly better BER performance. Compared with the SCL algorithm in [7, 8], the BP algorithm is superior when L = 1, equivalent when L = 2, but inferior when $L \geq 4$. However, for $L \geq 4$, the proposed BP

algorithm retains its advantage with significantly lower computational complexity and latency compared to SCL.

The algorithm can be implemented in Vitis Model Composer, tested in the Matlab environment, then can be synthesized into RTL by Xilinx Vivado and run on programmable logic. The implementation results indicate that high throughput and short latency can be achieved with a small amount of logic resources on Zynq UltraScale+ RFSoC DFE device XCZU67DR-2FSVE1156I. The results are summarized in the table 1 and table 2, including comparisons with several other methods [9, 13].

Table 1. *Encoder.*

Algorithms	LUTs	FFs	BRAM18Ks	DSPs	Latency (cycles)	F_{max} (MHz)
[9]	1486	7283	1	0	256	200
[13]	4518	8667	1	0	244	272
Proposed	425	460	1	0	10	622

Table 2. *Decoder.*

Algorithms	LUTs	FFs	BRAM18Ks	DSPs	Latency (cycles)	F_{max} (MHz)
[16]	13388	3688	15	0	262	106
[13]	1600	6507	3	0	244	102
Proposed	442	468	2	0	10	573

Table 1 and table 2 show that the proposed algorithm for the encoder and decoder uses significantly fewer hardware resources than previous methods, with a sharp reduction in the number of LUTs and FFs (only 425 LUTs, 460 FFs for the encoder and 442 LUTs, 468 FFs for the decoder). Meanwhile, the latency is only 10 cycles, 24 to 26 times lower, while the operating frequency (622 MHz, 573 MHz) is significantly higher than that of the compared methods, making it well-suited for 5G wireless systems. These improvements collectively contribute to enhanced performance and reduced hardware costs in real-world deployments.

4. CONCLUSIONS

In this paper, the authors developed a simplified equivalent algorithm for fully parallel encoding and decoding with the following features:

- The algorithm maintains equivalency with currently available parallel encoding and decoding methods while reducing computational complexity and minimizing processing latency;

- The algorithm is implemented in System Generator/Vitis Model Composer with parameterization capabilities, allowing easy maintenance and adaptation of the FPGA design to future standard changes.

- The design accuracy was validated through MATLAB simulations for all 3GPP codes, while RTL synthesis was performed using the compact and high-frequency capabilities of the Vivado tool, significantly enhancing latency and throughput on Xilinx UltraScale+ devices.

Acknowledgements: *This research is funded by Electric Power University under research project 2023, Grant Number: ĐTKHCN.10/2023*

REFERENCES

[1]. 3GPP TS 38.212, *"3GPP Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 15)"*, v15.2.0 (2018).

[2]. K. Niu and K. Chen, "*Stack decoding of polar codes*," Electronics Letters, Vol. 48, No. 12, pp. 695–696, (2012).

[3]. O. Afisiadis, et al., "*A low-complexity improved successive cancellation decoder for polar codes*," in 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, pp. 2116–2120, (2014).
[4]. K. Chen, K. Niu, and J. Lin, "*Improved successive cancellation decoding of polar codes*," IEEE Transactions on Communications, Vol. 61, No 8, pp. 3100–3107, (2013), Doi:10.1109/TCOMM.2013.070113.120993
[5]. I. Tal and A. Vardy, "*List Decoding of Polar Codes*," IEEE Transactions on Information Theory, Vol 61, No 5, pp. 2213–2226, (2015), DOI: 10.1109/TIT.2015.2410251.
[6]. M. Rowshan and E. Viterbo, "*Improved List Decoding of Polar Codes by Shifted-pruning*," 2019 IEEE Information Theory Workshop (ITW), Visby, Sweden, pp. 1-5, (2019). Doi: 10.1109/ITW44776.2019.8989330
[7]. Y. Peng; X. Liu; J. Bao, "*An Improved Segmented Flipped Successive Cancellation List Decoder for Polar Codes*," IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, (2020). Doi:10.1109/ccwc51732.2021.937593
[8]. B.Yuan; P. K. Keshab; "*Low-Latency Successive-Cancellation List Decoders for Polar Codes With Multibit Decision*," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 23, pp. 2268 -2280, (2014), Doi:10.1109/TVLSI.2014.2359793
[9]. H. Yoo, and I. C. Park, "*Partially Parallel Encoder Architecture for Long Polar Codes*," IEEE Transactions on Circuits Systems II Express Briefs, Vol. 62, no 3, pp.306-310, (2015). Doi: 10.1109/TCSII.2014.2369131
[10]. E. Arikan, "*Channel polarization: A method for constructing capacity Achieving codes for symmetric binary-input memoryless channels*," IEEE Trans. Inf. Theory, Vol. 55, No 7, pp. 3051-3073, (2009), Doi:10.1109/tit.2009.2021379
[11]. A. Arpure, and S. Gugulothu, "*FPGA implementation of polar code based encoder architecture*," International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, (2016), Doi: 10.1109/ICCSP.2016.7754231
[12]. M. Rowshan; et al., "*Logarithmic Non-uniform Quantization for List Decoding of Polar Codes*," 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)}, NV, USA, (2021). Doi:10.1109/ccwc51732.2021.9375932
[13]. G.Sarkis, et al., "*Flexible and Low-Complexity Encoding and Decoding of Systematic Polar Codes,*" IEEE Transactions on Communications, Vol. 64, No 7, pp. 2732 – 2745, (2016). Doi: 10.1109/TCOMM.2016.2574996
[14]. U. M. N. Raj, and E. V. Narayana, "*An advanced architecture with low complexity of partially parallel polar encoder*," International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, (2016). Doi: 10.1109/CESYS.2016.7889957.
[15]. Sarkis, Gabi et al., "*Fast Polar Decoders: Algorithm and Implementation",* IEEE Journal on Selected Areas in Communications, Vol.32, No. 5, pp. 946-957, (2014). Doi:10.1109/JSAC.2014.140514
[16]. UG897, *"Vivado design suite user guide: model-based dsp design using system generator"*, in: Xilinx User Guide, (2020).

TÓM TẮT

Thuật toán mã hóa và giải mã song song với kiến trúc phần cứng tối ưu cho mã Polar để giảm độ phức tạp và thời gian xử lý

Bài báo này trình bày việc phát triển một thuật toán tương đương đơn giản hóa để mã hóa và giải mã song song hoàn toàn mã Polar, giúp giảm độ phức tạp tính toán và độ trễ xử lý so với các phương pháp mã hóa và giải mã hóa song song hiện tại. Thuật toán được triển khai trong môi trường System Generator/Vitis Model Composer với khả năng tham số hóa, đảm bảo tính linh hoạt và dễ dàng điều chỉnh thiết kế FPGA để đáp ứng các tiêu chuẩn tương lai. Độ chính xác của thiết kế được kiểm chứng bằng mô phỏng MATLAB với mã chuẩn 3GPP. Ngoài ra, tổng hợp RTL thông qua Vivado đã cải thiện đáng kể cả độ trễ và thông lượng.

Từ khoá: Mã Polar; Mã hóa và giải mã song song; Kiến trúc phần cứng tối ưu.