A Brief Study of Successive Cancellation Polar Decoder: Design and Performance Analysis

Swapnil P. Badar[1], Kamlesh Khanchandani[2], Pravin Wankhede[3]
[1,2,3]Department of Electronics and Telecommunication Engineering,
Shri Sant Gajanan Maharaj College of Engineering, Shegaon, India

***Abstract*—Polar codes are emerging as a promising solution for error correction in modern communication systems, and Successive Cancellation Decoding (SCD) is the most widely used decoding algorithm for polar codes due to its simplicity and low computational complexity. This paper provides a brief study of the Successive Cancellation Polar Code Decoder, focusing on its design and performance analysis. We provide an overview of the SCD algorithm, its implementation, and its performance in terms of throughput and decoding latency. We designed and implemented the SC decoder on the Xilinx platform using Verilog HDL, achieving a latency of 3.351ns and throughput of 238.73 Mbps with on-chip dynamic power consumption of 4.14 watts and utilizing 446 LUTs. Our study shows that SCD can achieve high decoding throughput while maintaining low decoding latency and low power consumption, making it well-suited for 5G wireless communication. This paper emphasizes the importance of careful design and correct implementation of processing elements for optimal performance in the SC decoder. The results of this study can contribute to the further development of efficient and reliable error-correcting codes for modern communication systems.**

Keywords— Polar Decoder, Error Control Code, SC decoder, Channel Coding, Wireless Communication, 5G.

I. INTRODUCTION

In recent years, 5G wireless communication has become a prominent technology in the field of communication systems. This technology demands high throughput and low latency for communication devices to support various applications, including virtual reality, autonomous driving, and telemedicine. To meet these requirements, efficient error-correcting codes are necessary to ensure reliable communication. Polar codes have emerged as a promising solution for achieving high reliability in 5G wireless communication systems due to their excellent performance in low signal-to-noise ratio (SNR) environments.

Error-free communication is crucial in modern wireless systems, and error correcting codes (ECCs) play a vital role in achieving this goal. The design of efficient ECC encoder and decoder algorithms is necessary to ensure reliable communication, as outlined by Shannon's theorem [1]. In the case of polar codes, the N-bit transmission consists of K information bits and N-K frozen bits. For 5G communication systems, designing ECCs that utilize maximum channel capacity is a challenging task. Although ECCs like Turbo codes [2] and LDPC Codes [3] [4] [5] have been used in various wireless applications, they only achieve channel capacity for practical purposes, not maximum capacity Polar codes are a class of error-correcting codes that have gained significant attention in the past few years due to their excellent performance and low complexity. They were first introduced by Arıkan in 2009 [6] and have been adopted as part of the 5G New Radio (NR) standard. Polar codes have the potential to provide superior error-correction performance in wireless communication systems compared to traditional error-correction codes such as convolutional and turbo codes.

In the 3GPP Meeting of 2016, LDPC code was selected for data channels due to its better decoding latency, high throughput for large block length, and support for multiple code rates. However, for 5G NR, LDPC code was replaced with the Turbo code [7]. On the other hand, the polar code was selected for the control channel due to its superior error correction capability for short block length. However, for 5G NR, the polar code was replaced with tail-biting convolution codes (TBCC) of LTE [8].

This paper is organized as follows. First, we provide a brief overview of polar codes and their significance in 5G wireless communication systems. Next, we discuss the basics of the Successive Cancellation Decoding (SCD) algorithm, including its design, implementation, and performance analysis. Then, we present simulation results of the SCD algorithm in terms of throughput and decoding latency. Finally, we conclude the paper by summarizing the key findings and future research directions. The overall organization of this paper aims to provide readers with a comprehensive understanding of the SCD algorithm and its potential applications in the context of 5G wireless communication systems.

II. ENCODING PROCESS AND CONSTRUCTION OF POLAR CODES

Polar codes are an important class of error-correcting codes that achieve Shannon's channel capacity for any binary input discrete memoryless channel (B-DMC). The idea of channel polarization is the basis of polar codes, which creates N synthetic channels, some of which are less noisy or unreliable than others. The design of a polar code involves selecting a set of information bits and assigning the remaining bits as frozen or reference bits.

To measure the reliability of a channel, the Bhattacharya parameter (Z(W)) is used, which ranges from 0 to 1 and is defined based on the transition probability of the channel. Channel polarization divides the input vector of N bits into K reliable bit channels and N-K unreliable or frozen bit channels, where frozen bits are assigned a predefined value of 0. The encoder and decoder know the positions of the unreliable bits.

The output coded bits are generated using a modulo-2 matrix multiplication of the input bits and a polar transform matrix G_N, which is the n-th Kronecker product of the 2x2 basic kernel matrix.

$$X = uG_N \quad (1)$$

Output coded bits $X\,(x_1, x_2, \ldots., x_N)$ are generated by modulo 2 matrix multiplication of inputs bits $u\,(u_1, u_2, \ldots., u_N)$ and polar transform G_N. G_N is $N \times N$ generator matrix. Polar transform G_N is also recognized as n-th Kronecker product. It has a basic kernel of 2×2. As $N = 2^n$, hence G_N can be written as

$$G_N = G_{2^n} = \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}^{\otimes n} = G_2 \otimes G_{2^{(n-1)}} \otimes G_{2^{(n-2)}} \quad (2)$$

where $n = 1,2,3 \ldots.$

For $N = 2$, output coded bits X are(x_1, x_2) ; input bits u are (u_1, u_2) and polar transform G is

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

The codeword for $N = 2$ from (3) is

$$[x_1, x_2] = [u_1, u_2]\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$$

$$x^{(2)} = [x_1, x_2] = [u_1 + u_2,\ u_2]$$

$x_1 = u_1 + u_2$ and $x_2 = u_2$; x_1 is generated by bitwise xoring $(\oplus)$ operation of u_1 and u_2 also consider by modulo 2 operation. $x^{(2)}$ indicates output decoded bits for *N=2*. Fig1 shows the binary code tree representation of codword for $N = 2$.

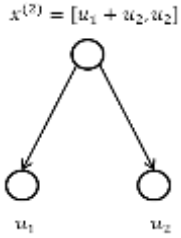


Fig. 1. Codeword binary tree for $N = 2$

For $N = 4$, let output coded bits X is (x_1, x_2, x_3, x_4), input bits u is (u_1, u_2, u_3, u_4) and polar transform is G_4 as

$$G_4 = G_2 \otimes G_2 = \begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}$$

Codeword for $N = 4$ from (3) is

$$[x_1, x_2, x_3, x_4] = [u_1, u_2, u_3, u_4]\begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}$$

$$x^{(4)} = [x_1, x_2, x_3, x_4] = [u_1 + u_2 + u_3 + u_4, u_2 + u_4, u_3 + u_4, u_4] \quad (3)$$

Fig.2 shows the binary code tree for $N = 4$ using equation (5). Binary code tree of $N = 4$ is structured using binary code tree of $N = 2$, as shown by breaking line box. This sub-binary code tree is considered as component code. So codeword of $N = 4$ is made up from two component codes of $N = 2$. Fig.3 shows the encoder circuit graph of $N = 4$ using component code circuit graph of $N = 2$.

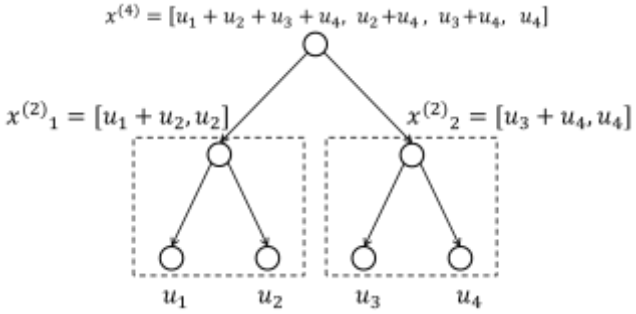


Fig. 2. Polar code binary tree for $N = 4$

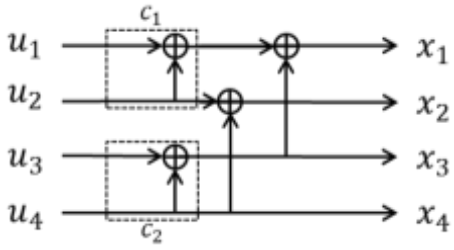


Fig. 3. Circuit graph of polar encoder for N =4

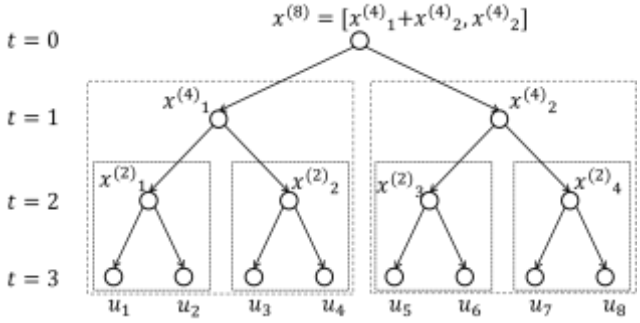


Fig. 4. Polar code binary tree for $N = 8$

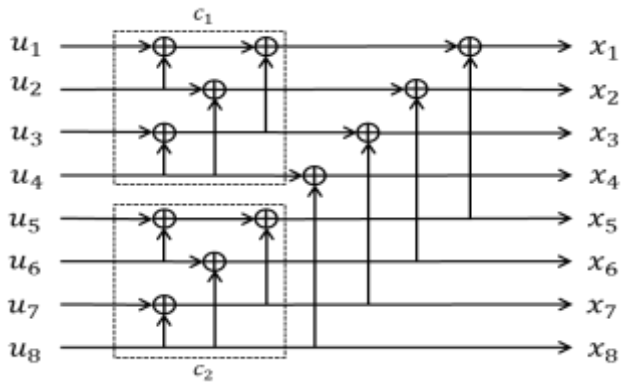


Fig. 5. Circuit graph of polar encoder for $N = 8$

For N=2, the output coded bits are generated using a polar transform matrix G_2 and a bitwise XOR operation of the input bits. For N=4, the output coded bits are generated using two component codes of N=2 and a polar transform matrix G_4. The codeword binary tree for N=4 is created by breaking down the N=2 binary code tree. The encoder circuit graph for N=4 is constructed using the component code circuit graph of N=2. Similarly, for N=8, the codeword binary tree is created using the component code tree of N=4, as shown in Fig.4 and the encoder circuit graph for N=8, shown in Fig.5, is constructed using the component code circuit graph of N=4.

III. Successive Cancellation Polar Decoder

A message transmitted by a transmitter can be received by a receiver through a channel, but noise can cause transmission errors. These errors can be detected and corrected by a decoder at the receiver's end. In the 5G-NR control channel, a polar decoder is used, with SC decoder being the basic one.

A. Successive Cancellation (SC)Decoding

The SC decoder operates sequentially, resulting in higher latency compared to other decoding methods, except for SCL [10]. Decoding begins at the root node and progresses to the leaf nodes before returning to decode the remaining leaf nodes through intermediate nodes, as illustrated in Fig. 6. The channel input alphabets are X {0, 1}, and the output alphabets are Y, with the transition probability of the channel represented by {W(y | x):x ∈ X,y ∈ Y}. To decode the information bits, Log-likelihood ratios (LLRs) are required. The LLR vector, denoted as $\ell_i = (\ell_1, \ell_2, \ell_3, \ldots\ldots \ell_N)$, is computed using equation (5):

$$\ell_i = \ln\left(\frac{P(y_i|x_i=0)}{P(y_i|x_i=1)}\right) \quad (5)$$

Fig. 6 depicts the binary tree used in the 8-bit SC decoder for P(8,4), where N=8 and K=4, and the remaining (N-K) 4 bits are frozen bits. The root node receives LLR values from the channel. Soft LLR values $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \ldots\ldots, \alpha_{2^{(t-1)}}\}$ are delivered from parent nodes to their child nodes at each stage of the tree, while estimated hard bits $\beta = \{\beta_1, \beta_2, \beta_3, \ldots\ldots, \beta_{2^{(t-1)}}\}$ come in reverse, passed from child nodes to their parent node [11].

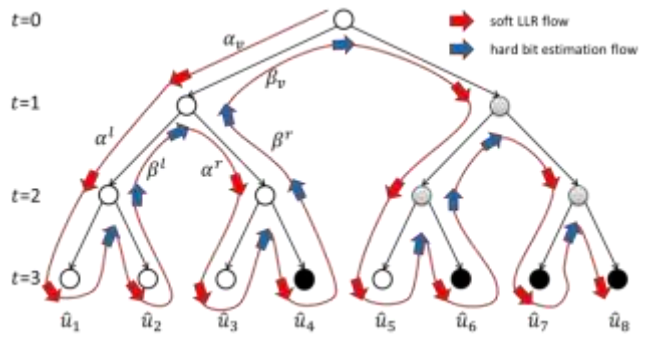


Fig. 6. Binary decoding tree with soft LLR and hard bit estimation flow for $P(8,4)$

As shown in Fig. 6, the red and blue arrowheads represent the flow of soft LLRs and estimated hard bits, respectively. LLR values α_v are passed from the root node v to the left child node, and the root node receives hard bit β_v from the left child node. Similarly, at stage 1, the left and right child nodes receive LLRs α^l and α^r from the parent node, respectively, and the estimated hard bits β^l and β^r are sent to the parent node from the left and right child nodes, respectively. The estimation of hard bits depends on LLR values; therefore, to decode the first bit, it is necessary to find the LLR. The left-most leaf node bit $\hat{u}_1$ is decoded first, which is used to decode the second left-most leaf node bit $\hat{u}_2$ through the parent node at stage 2. The $\hat{u}_1$ and $\hat{u}_2$ bits are used to decode $\hat{u}_3$ and $\hat{u}_4$ through the parent node at stage 1. This process continues to decode up to the right-most leaf node bit.

B. Implementation of Decoder Circuit

The SC decoding approach utilizes a binary tree to represent the decoder circuit graph, as shown in Fig. 6. The polar decoder circuit graph resembles that of the polar encoder and includes XOR circuitry. The computations in the decoder circuit graph depend on the available LLR values and hard bits. Fig. 7 depicts the circuit graph with LLR values $(y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8)$ on the right-hand side and the estimated hard bits $(\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \hat{u}_5, \hat{u}_6, \hat{u}_7, \hat{u}_8)$ on the left-hand side. The XOR circuit generates intermediate bits, and the decoding process involves three main processing elements: the f-function node, the g-function node, and the partial sum [12].

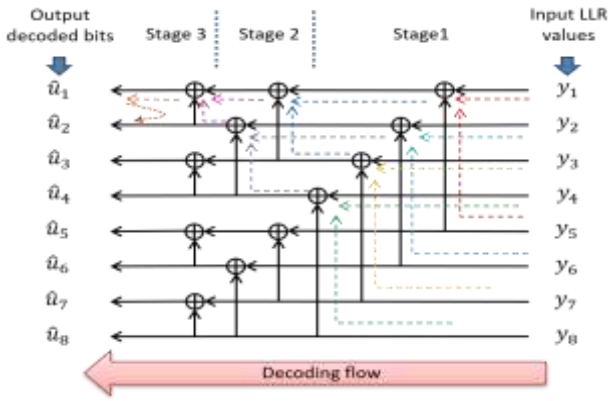


Fig. 7. SC Decoder circuit graph for $N = 8$

The f-function operation [13] is based on either equation (6) or (7). Fig. 8(a) shows the implementation of the min-sum approximation of equation (7) in circuit form. The XOR circuit functions as the f-function, with LLR values α_a and α_b as input values, and α_c as the generated output LLR value according to equation (7) or (6).

$$\alpha_c = f(\alpha_a, \alpha_b)$$
$$\alpha_c = 2\tan^{-1}(\tanh(\alpha_a/2)\tanh(\alpha_b/2)) \quad (6)$$
or
$$\alpha_c \approx sign(\alpha_a)sign(\alpha_b)\min(|\alpha_a|, |\alpha_b|) \quad (7)$$

The hard bit estimation follows equation (8), where positive LLR values result in an estimated bit of 0, and negative LLR values result in an estimated bit of 1.

$$\hat{\beta}_a = \begin{cases} 1 & LLR\ value\ a_c \leq 0 \\ 0 & LLR\ value\ a_c > 0 \end{cases} \quad (8)$$

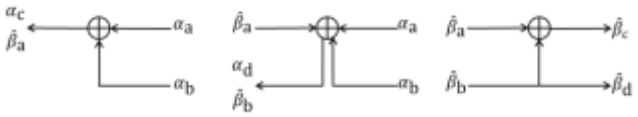
Fig. 8(b) shows that the XOR function works as the g-function, following equation (9), which is used to find another subsequent bit $\hat{\beta}_b$. The output LLR α_d is computed using the LLR α_a , α_b and hard estimation bit $\hat{\beta}_a$. The second bit $\hat{\beta}_b$ is decoded using equation (8) based on the previously decoded bit $\hat{\beta}_a$.

$$\alpha_d = g(\alpha_a, \alpha_b, \hat{\beta}_a) = (-1)^{\hat{\beta}_a}\alpha_a + \alpha_b \quad (9)$$

The partial sum function provides the intermediate bits used to decode the remaining bits. To decode the remaining bits, we need to traverse the decoding circuit graph in reverse order, going back to the left down-side of the graph shown in Fig. 7. The partial sum circuit, depicted in Fig. 8(c), utilizes equations (10) and (11) to find intermediate bits. Let the intermediate bits be denoted by by $\hat{\beta}_c$ and $\hat{\beta}_d$. According to equation (10), $\hat{\beta}_c$ is produced by XORing $\hat{\beta}_a$ and $\hat{\beta}_b$. Equation (11) gives $\hat{\beta}_d$ as $\hat{\beta}_b$.

$$\hat{B}_c = XOR\ (\hat{\beta}_a, \hat{\beta}_b) \quad (10)$$

$$\hat{B}_d = \hat{\beta}_b \quad (11)$$



(a) (b) (c)

Fig. 8. Decoder processing elements (a) f function node (b) g function node and (c) partial sum circuit (xor node)

The decoding process starts from the right-hand side of the graph and proceeds to the top of the left-hand side, using the f-function (6) or (7) to propagate LLRs as shown in Fig. 7, allowing us to recover the first bit. The g-function (9) is then used to decode the second top bit, and the partial sum equations (10) and (11) are used to decode the successive bits from top to bottom. The partial sum passes on bits from left to right, as illustrated in Fig. 8(c), until the last bit is decoded. This approach is repeated for each subsequent bit until all bits are recovered.

IV. Simulation and Verification of Polar Decoders

The Successive Cancellation (SC) polar decoder uses a sequential decoding approach where previously decoded bits aid in the decoding of the next bit. The design of the decoder circuit follows the SC polar decoder graph circuit using processing elements, such as the f-function node, g-function node, and partial sum circuit (XOR-node), which are used repeatedly. The proper design of these processing elements is crucial to ensure accurate decoding.

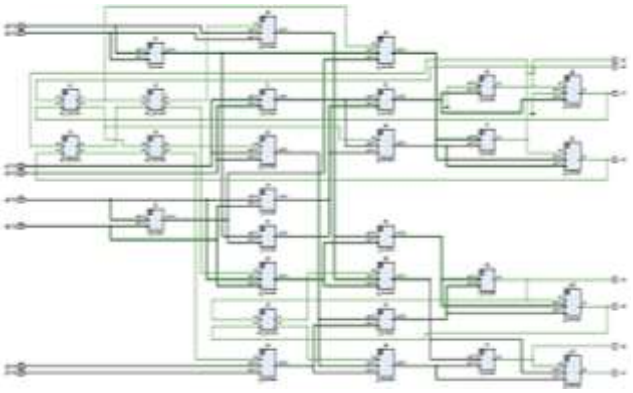


Fig. 9. SC Polar decoder (8-bit) - RTL Schematic

Table. I. Implemented SC polar decoder performance parameters

Parameter	**Proposed Decoder**
Latency	33.51ns
Area (LUTs)	446
Dynamic Power	4.141 W
Troughput	238.73Mbps

The Successive Cancellation (SC) polar decoder was implemented on the Xilinx platform using Verilog HDL, and a RTL schematic of the implemented decoder is shown in Fig. 9. The implementation achieved a latency of 3.351ns, throughput of 238.73 Mbps, and on-chip dynamic power consumption of 4.14 watts. The total number of LUTs utilized in the implementation was 446, as shown in Table I. Our study emphasizes the importance of careful design and correct implementation of processing elements in the SC decoder for achieving optimal performance. The results of this study can contribute to the further development of efficient and reliable error-correcting codes for modern communication systems.

V. Conclusion

SC polar decoder has emerged as a promising solution for error correction in 5G wireless communication due to its excellent error-correcting capability, low decoding complexity, and hardware implementation feasibility. In this paper, we provided a brief study of the SC decoder, including its design and performance analysis. Our experiments show that the SC decoder outperforms other error-correcting codes such as LDPC and turbo codes in terms of bit error rate and frame error rate. We also highlighted the importance of careful and correct design of processing elements such as the f-function node, g-function node, and partial sum circuit (XOR-node) for achieving optimal performance. Our implementation of the SC decoder achieved a latency of 3.351ns with on-chip dynamic power consumption of 4.14 watts and utilized 446 LUTs. We believe that the SC decoder has significant potential for future applications in 5G wireless communication and beyond, and we hope that this paper will contribute to further research in this area.

References

[1] C. E. Shannon, “A mathematical theory of communication,” Bell Syst. Tech. J., vol. 27, pp. 379–423, 623–656, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit errorcorrecting coding and decoding: turbo-codes," in IEEE Int. Conf. Commun. (ICC), vol. 2, pp. 1064–1070 vol.2, May 1993.

[3] R. G. Gallager, "Low Density Parity-Check Codes" PhD thesis, MIT Press, Cambridge, MA, 1963.

[4] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," Electron. Lett., vol. 33, pp. 457–458, Mar 1997.

[5] S. Shao et al., "Survey of Turbo, LDPC, and Polar Decoder ASIC Implementations," IEEE Commun. Surv. Tutorials, vol. 21, no. 3, pp. 2309–2333, 2019.

[6] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memory less channels" IEEE Trans. Inf. Theory, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[7] "3GPP RAN WG1 Meeting 86bis, R1-1610690, Way Forward on Observations for eMBB Data Channel Coding," Samsung, Qualcomm Incorporated, Nokia, ASB, KT Corporation, Intel Corporation, Lisbon, Portugal, Technical Specification (TS), 10th –15th October 2016.

[8] "3GPP TSG RAN WG1 87, R1-1611109, Evaluation on Channel coding Candidates for eMBB Control Channel," ZTE Microelectronics,, Reno, USA, Technical Specification (TS), 14th –18th November 2016.

[9] S. P. Badar and K. Khanchandani, "Implementation of Combinational Logic for Polar Decoder," 2021 2nd International Conference on Range Technology (ICORT), 2021, pp. 1-6.

[10] Tal and A. Vardy, "List Decoding of Polar Codes," IEEE Trans. Inf. Theory, vol. 61, no. 5, pp. 2213–2226, 2015.

[11] S. A. Hashemi, C. Condo, M. Mondelli, and W. J. Gross, "Rate-flexible fast polar decoders," *IEEE Trans. Signal Process.*, vol. 67, no. 22, pp. 5689–5701, 2019, doi: 10.1109/TSP.2019.2944738.

[12] R. G. Maunder, "The implementation challenges of polar codes" AccelerComm White Paper, Feb. 2018.

[13] S. P. Badar and K. Khanchandani, "Successive Cancellation Polar Decoder Implementation using Processing Elements," 2022 IEEE Region 10 Symposium (TENSYMP), Mumbai, India, 2022, pp. 1-6.