electronics

Article

Parity-Check-CRC Concatenated Polar Codes SSCFlip Decoder

Qasim Jan [1,2,3], Shahid Hussain [4], Muhammad Furqan [5], Zhiwen Pan [1,2,*], Nan Liu [1] and Xiaohu You [1,2]

[1] National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China
[2] Purple Mountain Laboratories, Nanjing 211100, China
[3] Department of Computer Science, COMSATS University Islamabad, Attock Campus, Attock 43600, Pakistan
[4] School of Medicine, Computer Science and Engineering, University of Galway, H91 TK33 Galway, Ireland
[5] Department of Computer Science, Women University Swabi, Swabi 23430, Pakistan
* Correspondence: pzw@seu.edu.cn

Abstract: Successive cancellation flip decoding requires a large number of extra successive cancellation decoding attempts at low signal-to-noise ratios (SNRs), resulting in high decoding complexity. In addition, it has a long decoding latency. Although modifications have been proposed in successive cancellation flip decoding, these still have high computational complexity at low SNRs due to a huge number of additional successive cancellation decoding attempts. It is desirable to detect the unsuccessful successive cancellation decoding process at an early stage in the additional successive cancellation flip attempts and stop it that can reduce the decoding complexity. This paper combines the parity-check-CRC concatenated polar codes with the low-latency simplified successive cancellation decoding and proposes a parity-check-CRC concatenated polar codes simplified successive cancellation flip (PC-CRC-SSCFlip) decoder. It further employs the parity-check vector to identify the unsuccessful simplified successive cancellation flip decoding at an early stage and terminates so that it can minimize the decoding complexity on average. Additionally, this work proposes an error-prone flipping list by incorporating the empirically observed indices based on channel-induced error distribution along with the first bit of each Rate-1 node. The proposed technique can identify more than one error-prone bit through a flipping list and correct them. In addition, the parity-check vector further narrows down the search space for the identification of erroneous decisions. Simulation results show that 60% of unsuccessful additional successive cancellation decoding attempts terminate early rather than decode the whole codeword. The proposed PC-CRC-SSCFlip decoder has approximately 0.7 dB and 0.3 dB gains over successive cancellation and successive cancellation flip decoders, respectively, at a fixed block error rate (BLER) = 10^{-3}. Additionally, it reduces the average computational complexity and decoding latency of the successive cancellation flip decoder at low-to-medium SNRs while approaching successive cancellation decoding complexity at medium-to-high SNRs.

Keywords: bit flip; parity-check vector; polar codes; successive cancellation; SCFlip

Citation: Jan, Q.; Hussain, S.; Furqan, M.; Pan, Z.; Liu, N.; You, X. Parity-Check-CRC Concatenated Polar Codes SSCFlip Decoder. *Electronics* **2021**, *11*, 3839. https://doi.org/10.3390/electronics11233839

Academic Editors: Francesca Vatta, Roberto Garello and Gianluigi Liva

Received: 5 June 2022
Accepted: 17 November 2022
Published: 22 November 2022

1. Introduction

Polar codes have the ability to achieve Shannon capacity over the binary discrete memoryless channel with infinite block-length ($\mathcal{N}$) using the successive cancellation (SC) decoder with computational complexity $O(\mathcal{N}\log\mathcal{N})$ [1]. However, the BLER performance of SC cannot compete with the low-density-parity-check (LDPC) and turbo codes with finite block-length [2,3]. Successive cancellation list (SCL) [4] and successive cancellation stack (SCS) [5] decoders were proposed to improve the error correction performance of finite block-length. The SCL runs multiple decoders in parallel and maintains a list of candidate messages, but it has a high computational and space complexities of $O(\mathcal{L}\mathcal{N}\log\mathcal{N})$ and $O(\mathcal{L}\mathcal{N})$, where $\mathcal{L}$ is the number of lists in the SCL. Moreover, SCS traverses over an ordered decoding tree for identifying the optimal path in a set of candidate paths. The computational and space complexities of the SCS are $O(\mathcal{D}\mathcal{N}\log\mathcal{N})$ and $O(\mathcal{D}\mathcal{N})$, where $\mathcal{D}$ is the number of candidate paths. The authors in [6,7] further concatenated the CRC with polar codes and

proposed the CRC-aided (CA)-SCL decoding algorithm, which significantly improves the error correction performance of SCL. The work in [8] concatenated the parity check with polar codes as an outer code to improve the error correction performance of SCL decoding. The parity check in [8] helps the SCL decoder prune the error paths promptly. However, the SCL decoder has a high complexity, decoding latency, and resource consumption compared with standard SC decoding.

Afisiadis et al. [9] proposed an alternative technique, namely successive cancellation flip (SCFlip), that improves the error correction performance of polar codes while keeping average computational complexity close to the standard SC decoder at wide SNRs. The SCFlip flips a single bit, and its error correction performance is restricted to the CA-SCL with $\mathcal{L} = 2$ at a high SNRs. However, at low SNRs, it needs a large number of extra SC decoding attempts and results in high computational complexity and decoding latency. The authors in [10] proposed an optimized metric to determine the first wrongly estimated bit and introduce the nested flip concept. To narrow down the search space for the identification of the first erroneous bit, Zhang et al. in [11] proposed a critical set (CS) and progressive bit-flipping decoding, which considers all Rate-1 nodes level by level. The fast-SC decoding [12] was merged with SCFlip decoding in [13] to minimize the average execution time of the SCFlip decoding. The authors in [14,15] further extended the fast simplified successive cancellation flip (fast-SSCFlip) for Type-II, Type-III, Type-IV, and Type-V nodes [16] that provides a better decoding speed compared with the fast-SSCFlip decoder. Furthermore, the partitioned SCFlip (PSCF) subdivides the codeword into partitions and performs SCFlip decoding on each partition [17]. Wang et al. proposed multiple bit-flipping concepts for the fast simplified successive cancellation flip (fast-SSCFlip) decoding [18]. They also merged the partitioned SCFlip decoding [17] with fast-SSCFlip decoding and proposed a partitioned fast-SSC-flip (PA-fast-SSCFlip) decoding algorithm. The work in [19] investigated the distributions of channel-induced errors and proposed an improved SCFlip decoding for low-rate codes based on the observation of channel-induced errors. Chandesris et al. proposed dynamic SCFlip (D-SCFlip) decoding to flip one or several unfrozen bit decisions [20]. A threshold SCFlip (TSCF) decoding algorithm was proposed to improve the BLER performance of the SC decoder [21]. A multi-CRC polar codes construction algorithm was introduced to correct an error in time and slow down the error propagation [22]. A simplified dynamic SCFlip polar decoding was proposed in [23], which replaced the logarithmic and exponential calculation of [20] with simple approximation. The distributed parity checks along with CRC was used to improve the BLER performance of the standard SC-flip decoder [24]. Error-aware SCFlip decoding was introduced in [25] for distributed CRC polar codes, which minimizes the decoding latency. Recently, the work in [26] merged the partitioned SC-flip idea with high-speed SC-flip decoders and proposed an improved partitioned fast-SSC-flip (PFSSC-flip) decoder. Inspired by the SCFlip decoding algorithms, the authors in [27–29] further extended the bit-flip algorithm for belief propagation (BP) decoding.

The SCFlip decoding algorithm and its variants significantly improved the BLER performance with the average complexity of the SC decoder at high SNRs. However, they require many extra decoding iterations at low SNR regions, which results in a high decoding complexity and long latency. Consequently, this work proposes the PC-CRC-SSCFlip decoder for polar codes. The proposed PC-CRC-SSCFlip decoder improves the error correction performance and minimizes the decoding complexity and latency of the SCFlip decoder. The key idea of this work is to identify the first channel-induced error at an early stage in the SSCFlip decoder and correct it as efficiently as possible. Additionally, this work proposes an error-prone flipping list (FL) by incorporating the empirically observed indices based on channel-induced error distribution along with the first bit of each Rate-1 node. The proposed FL narrows the search space for the identification of the first erroneous bit by utilizing the parity-check-based mechanism.

Simulation results show that the proposed PC-CRC-SSCFlip stops the corresponding decoding process whenever a parity check detects an error and thus avoids unnecessary de-

coding steps. The improved PC-CRC-SSCFlip decoder has approximately 0.7 dB and 0.3 dB gains over the SC and SCFlip decoder, respectively, at a fixed BLER = 10^{-3}. Moreover, it reduces the average computational complexity and decoding latency of the SCFlip decoder at low-to-medium SNRs while approaching the SC decoding complexity at medium-to-high SNRs, i.e., $O(\mathcal{N}\log\mathcal{N})$.

The contribution of this work is as follows:

- We extend the work of [24] for simplified successive cancellation decoding and propose a PC-CRC-SSCFlip that exploits the role of the distributed parity checks to detect the incorrectly estimated erroneous bit at the early stages in the SSCFlip decoding and corrects that bit. The proposed technique minimizes the average decoding complexity and latency by halting the decoding process with early detection.
- We introduce an error-prone flipping list by incorporating the indices that are more suspected to be in an error [19] and the first bit of each Rate-1 node [11]. The proposed algorithm recalls the parity-bit and narrows down the search space whenever an error is detected. Once an error is detected by a parity bit, the proposed technique considers the bits of FL preceding that parity bit.
- In addition, the proposed PC-CRC-SSCFlip can correct two erroneous bits. It flips the first erroneous decision generated by the initial SC decoding, while the second erroneous decision is flipped in the decoding trajectory determined by the previous flips. This algorithm can be easily employed for more than two erroneous decisions. Although the multi-erroneous decision flipping algorithm improves the BLER performance of SCFlip decoding, it results in high computational complexity. The PC-CRC-SSCFlip immediately terminates the corresponding process with the error detection that minimizing the average decoding complexity and latency of the SCFlip.

The work in [24] introduced the distributed parity check method for the standard SCFlip decoder. Nevertheless, the proposed work extends the distributed parity check method for the high-speed successive cancellation (i.e., the simplified successive cancellation (SSC)) decoder, which has a smaller decoding latency compared with the standard SCFlip decoder. Additionally, the work in [24] uses the absolute value of the log-likelihood ratio (LLR) with an additional penalty factor according to the location of the first failed parity check. In contrast, the proposed work incorporates the empirically observed probabilities of channel-induced errors [19] with the critical set (CS) [13] and introduces a predetermined error-prone flipping list (FL) for the identification of incorrectly estimated bits at the start of the SC decoding, thus leading to complexity saving.

The proposed algorithm achieves a BLER performance much better than the conventional SC, SCFlip decoding, and approaches to the lower bound of oracle-assisted (OA)-SCFlip-2.

The rest of this paper is organized as follows: Section 2 briefly explains the concept of polar codes encoding, SC, SSC, SCFlip, and OA-SCFlip decoding. Section 3 presents a detailed description of the proposed technique. Section 4 discusses the simulation result in detail, and Section 5 concludes the paper.

2. Preliminaries

2.1. Polar Codes and SC Decoding

Erdal Arikan introduced polar codes in [1] to achieve channel capacity, which splits the $\mathcal{N}$ number of channels into $\mathcal{K}$ reliable and $\mathcal{N}-\mathcal{K}$ unreliable channels. The message bits are transmitted on the most $\mathcal{K}$ reliable channels. These $\mathcal{K}$ reliable channels are known as information bit channels, and the $\mathcal{N}-\mathcal{K}$ unreliable channels are referred to as frozen bit channels. A polar code of length $\mathcal{N}=2^n$ and dimension $\mathcal{K}$ is denoted as $P(\mathcal{N},\mathcal{K},\mathcal{A})$ with rate $\mathcal{R}=\mathcal{K}/\mathcal{N}$, where $\mathcal{N}$ is the block-length, $\mathcal{K}$ is the message-length, $\mathcal{A}\subset\{1,\cdots,\mathcal{N}\}$ is the set of indices of information bits. The indices of frozen bits are the set of a complement

of $\mathcal{A}$ ($\mathcal{A}^c$) that is known to the transmitter and receiver. In the polar codes, the message is encoded through multiplication with the generator matrix as follows:

$$x_1^{\mathcal{N}} = u_1^{\mathcal{N}} \mathbf{G}^{\otimes n} \tag{1}$$

where $x_1^{\mathcal{N}} = (x_1, x_2, \cdots, x_{\mathcal{N}})$ represents the encoded codeword vector, $u_1^{\mathcal{N}} = (u_1, u_2, \cdots, u_{\mathcal{N}})$ is the input source vector, and $\mathbf{G}^{\otimes n}$ is the Arikan kernel, which is the n-th Kronecker product ($\otimes$) of $\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

The conventional SC decoding algorithm traverses over a depth-first binary tree whilst prioritizing the left branch. Figure 1 exemplifies an SC decoding binary tree with $\mathcal{N} = 8$ and $\mathcal{R} = 0.5$, where the channel LLR $\big(L(y)\big)$ is initialized at the root node. $L(y)$ denotes the channel LLR vector, i.e., $L(y_1^{\mathcal{N}}) = \{L(y_1), \cdots, L(y_{\mathcal{N}})\}$. Each internal (non-leaf) node at level L computes the left $L(y^l) = \{L(y_1^l), \cdots, L(y_{\frac{\mathcal{N}}{2^L}}^l)\}$ and right $L(y^r) = \{L(y_1^r), \cdots, L(y_{\frac{\mathcal{N}}{2^L}}^r)\}$ LLR vectors and assigns them to child nodes following (2) and (3) for further processing.

$$\begin{aligned} L(y_i^l) = \text{sign}(L(y_i))\text{sign}(L(y_{i+\mathcal{N}/2^L})), \\ \min\big(|L(y_i)|, |L(y_{i+\mathcal{N}/2^L})|\big) \end{aligned} \tag{2}$$

$$L(y_i^r) = L(y_{i+\mathcal{N}/2^L}) + (1 - 2u_i^l)L(y_i). \tag{3}$$

where u is the partial sum vector of the node at the L-th level and computed as:

$$\begin{cases} u_{2i-1} = u_i^l \oplus u_i^r, \\ u_{2i} = u_i^r, \qquad for \quad 1 \le i \le \mathcal{N}/2^L. \end{cases} \tag{4}$$

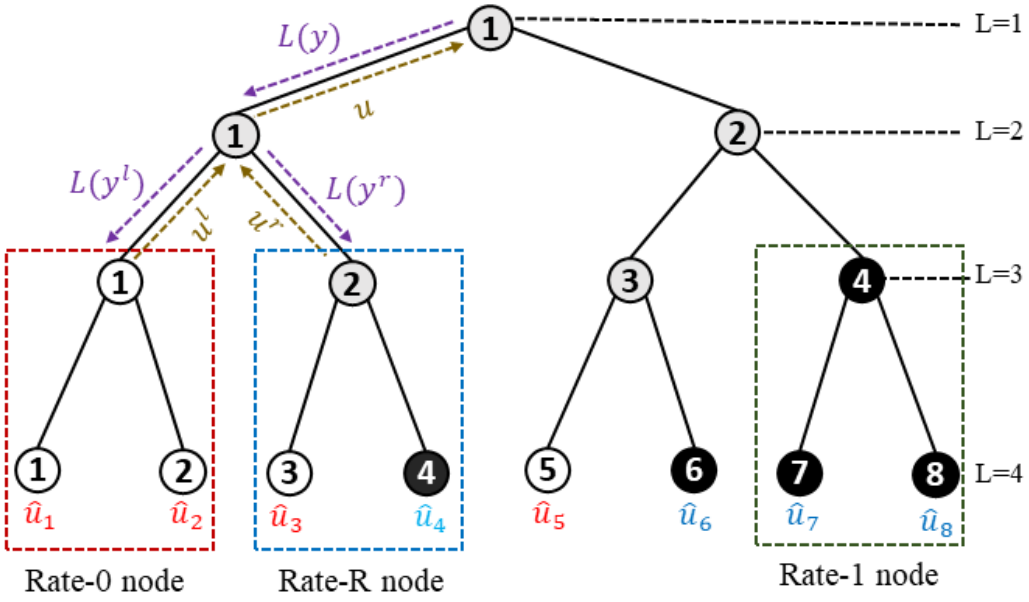

Figure 1. The binary tree representation of the SC decoding at block-length $\mathcal{N} = 8$ and $\mathcal{R} = 0.5$.

The $u^l = \{u_1^l, \cdots, u_{\mathcal{N}/2^L}^l\}$ and $u^r = \{u_1^r, \cdots, u_{\mathcal{N}/2^L}^r\}$ are the partial sum vectors resulting from the left and right offspring. The estimated bit $\hat{u}_i$ can be obtained at the leaf node based on the soft-decision LLR ($L(\hat{y}_i)$) using (5).

$$\hat{u}_i = \begin{cases} 0 & i \notin \mathcal{A}, \\ \frac{1-sign\left(L(\hat{y}_i)\right)}{2} & i \in \mathcal{A}. \end{cases} \tag{5}$$

2.2. Simplified Successive Cancellation (SSC) Decoding

The conventional successive cancellation algorithm has high decoding latency due to the serial nature of the SC decoder. Alamdar-Yazdi and Kschischang proposed simplified successive cancellation decoding to mitigate the decoding latency without altering the BLER performance of the SC decoder [30]. They only activate and decode the constituent codes with Rate-1 nodes while setting the partial sum vector (u) of a Rate-0 node to the zero codewords activating its descendants.

Figure 1 visualizes SSC decoding for $P(8,4)$, where Rate-0, Rate-R, and Rate-1 constituent codes are encircled in red, blue, and green boxes, respectively. In the SSC decoder, the constituent codes are referred to as Rate 0 nodes if all their descendants are frozen bits. The constituent codes are known as Rate-1 nodes if all their descendants are information bits. A node that contains both frozen and information bits as descendants are called the Rate-R constituent code. In Figure 1, white, grey, and black circles represent the Rate-0, Rate-R, and Rate-1, respectively.

2.3. Successive Cancellation Flip (SCFlip) Decoding

Successive cancellation algorithm decodes the codeword sequentially, and the previously estimated bit ($\hat{u}_i$) plays an important role in the decoding of the remaining undecided bits. Thus, the preceding incorrectly estimated bit ($\hat{u}_i$) can lead to error propagation. Consequently, the SCFlip decoder was proposed to identify and correct the first incorrectly estimated bit [9]. Initially, it estimates the message by performing SC decoding in which the C-bits CRC is used to verify the message ($\hat{u}_1^{\mathcal{N}}$). The decoding process is considered successful when the CRC is passed. Otherwise, SCFlip makes a list of the least reliable hard decisions. The list contains the $\mathcal{T}$ number of smallest absolute soft-decision LLR $|L(\hat{y}_1^{\mathcal{N}})|$ of information bits and is used to identify the error-prone information bit to be flipped in subsequent attempts. An C-bits CRC is used to validate the estimated message at each additional attempt. The decoding process is terminated if the CRC is verified or the $\mathcal{T}$ additional attempts reach the limit. The error correction performance of SCFlip at high SNR is limited to the CA-SCL with a small list size (i.e., $\mathcal{L} = 2$) with the flipping of a single bit. Additionally, at low SNRs, this results in high computational complexity due to a large number of extra SC decoding attempts. In consequence, this work proposes PC-CRC-SSCFlip decoding for polar codes that improves the error correction performance and minimizes the average decoding complexity and latency.

2.4. Oracle-Assisted Successive Cancellation (OA-SC) Decoding

In SC decoding, the error propagation is always caused by the erroneous decisions that occur due to the channel noise [9]. Oracle-assisted SC (OA-SC) decoding was proposed to characterize the impact of the channel-generated error by maintaining the foreknowledge of the transmitted information block ($u_1^{\mathcal{N}}$) [9]. The OA-SC corrects the erroneous decision at the decision level and ensures the restrictions of the error propagation. Moreover, it keeps track of the number of channel-generated errors. It was further proven by [10] that the OA-SC decoder provides optimal frame error correction performance of SC-flip and can serve as a lower bound for evaluating the error correction performance of practical SC-flip decoders. In the sequel, we consider the OA-SC BLER performance as a lower bound for validating the performance of the PC-CRC-SSCFlip decoding.

3. Proposed Scheme

This section provides a detailed description of the construction of FL. Moreover, it presents a detailed encoding and decoding mechanism of the proposed technique.

3.1. Construction of FL

The predetermined FL contains the most error-prone bit indices and narrows down the search space for the identification of incorrectly estimated bits. The detailed methodology for the construction of FL is given as elaborated upon below.

Zhang et al. in [11] proved through Gaussian approximation that the first index of each Rate-1 node is more likely to incur an error than other bit indices and proposed a critical set (CS) for the identification of incorrectly estimated bits. They divided the entire polar codes into multiple sub-polar codes and viewed the SC decoding sub-block by sub-block perspective. The sub-block includes only non-frozen bits and has a coding rate of $\mathcal{R} = 1$.

For instance, in Figure 2, node E is sub-polar codes with the coding rate $\mathcal{R} = 1$ and treated as a sub-block. It has four non-frozen bits, i.e., $\{u_{13}, u_{14}, u_{15}, u_{16}\}$. Nodes D and B include two non-frozen bits. Moreover, nodes A and C have a single non-frozen bit u_6 and u_{10}, respectively. Both nodes A and C are viewed as special sub-blocks because they are themselves treated as the root node and leaf node. They proved through observation that if the sub-block decoding output is incorrect, then the first non-frozen bit in this sub-block has a high probability of being estimated to be incorrect as well. Consequently, they collected the first non-frozen bit of each sub-block and proposed a new flipping set that is known as CS. The first bit index of each sub-block is fenced in the blue rectangle in Figure 2. For instance, in Figure 2, there are five sub-blocks, i.e., A, B, C, D, and E. Hence, the first bit of each sub-block is added to CS and CS = $\{u_6, u_7, u_{10}, u_{11}, u_{13}\}$.

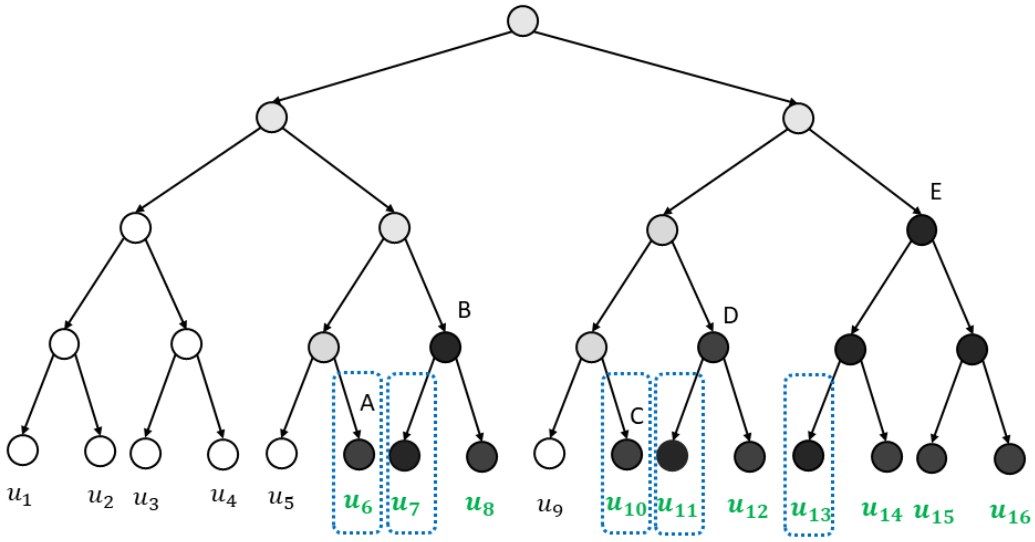

Figure 2. Rate-1 node presentation through SC binary tree for polar codes at $\mathcal{N} = 16$ and $\mathcal{K} = 10$.

Another independent approach based on the distribution of first channel-induced errors was proposed for the identification of incorrectly estimated bits [19]. It was proven in [19] that a few non-frozen bit indices have a high error probability and can be employed to determine the wrong decisions. Following [19], we perform the simulation for 5×10^6 blocks for $P(512, 256)$ under different SNR regions and determine the frequency occurrence of more than one channel-induced error. The simulation results in Figure 3 show the frequency occurrence of the channel induced-error E_1, E_2, and E_3. It can be seen that the SC decoding fails due to a single channel induced-error, i.e., E_1. Additionally, the frequency occurrence of E_1 has a linear relationship with SNR and E_2 and E_3 have an inverse relationship with SNR, i.e., when SNR increases, the E_1 frequency occurrence increases while the frequency occurrences of E_2 and E_3 decrease.

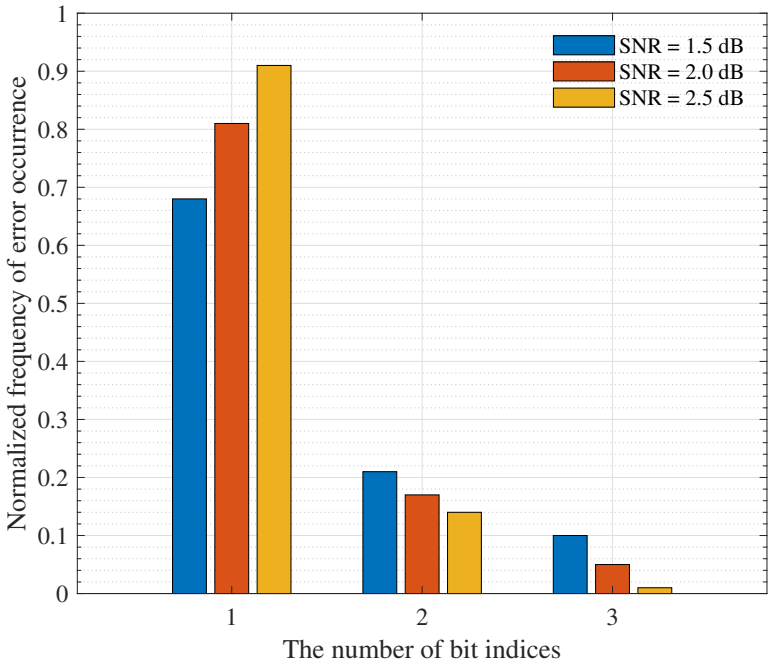

Figure 3. Channel-induced error occurrence frequencies under various SNR regions for $P(512, 256)$.

The above discussion concludes that most of the time, SC decoding fails due to the first channel-induced error. In addition, the first-bit index of the Rate-1 node [11] and the bit indices empirically observed by simulation [19] have higher error probabilities. The proposed work incorporates the empirically-observed probabilities of channel-induced errors with CS and proposes a predetermined error-prone FL for the identification of incorrectly estimated bits. The proposed FL is constructed according to Equation (6).

$$\text{FL} = \text{CS} \cup E_{1,2} \tag{6}$$

For instance, in Figure 2, if the flipping list of [19] contains $\{u_6, u_8\}$, then the proposed work takes the union of it with CS and constructs FL = $\{u_6, u_7, u_8, u_{10}, u_{11}, u_{13}\}$.

3.2. Parity-Check-CRC-Concatenated Polar Encoder

The proposed mechanism partitions the encoding process into two parts, i.e., the outer code and inner code. The outer code consists of two layers: parity check and CRC encoder. In the first layer, the message vector $u_1^{\mathcal{M}} = \{u_1, u_2, \cdots, u_{\mathcal{M}}\}$ is encoded through the parity function as an outer code. The parity-check encoder partitions the message vector $u_1^{\mathcal{M}}$ into segments according to the cardinality of the parity-check vector and calculates the parity bit for each segment. Each parity bit is then appended at the end of the corresponding segment. The resulting partitioned segments may either be the same or distinct sizes. Figure 4 demonstrates the underlying mechanism of the parity check encoder that partitions the message vector into equal segments. It appends a parity bit to the end of each of the corresponding segments such that a parity bit is represented by w_{p_k}-bits. Each parity bit relies on the preceding non-frozen bit indexes and is calculated according to (7).

$$w_{p_k} = \left(\bigoplus_{i \in \mathcal{A}_k, i \neq p_k} u_i \right). \tag{7}$$

where the $\bigoplus$ represents the module-2 sum operation, $\mathcal{A}_k \subset \mathcal{A}$ is the set of preceding non-frozen bit indices, and p_k denotes the index of k-th parity bit such that $p = (p_1, p_2, \cdots, p_K)$, and $p \subset \mathcal{A}$.

In the proposed concatenated PC-CRC-polar coding scheme technique, the parity-check encoder generates a codeword as an outer code with length $\mathcal{P}$, such that $\mathcal{P} = \mathcal{M} + K$ as shown in Figure 5. The resultant vector $w_1^{\mathcal{P}} = u_1^{\mathcal{M}+K}$ is then assigned to the second layer and processed through the CRC encoder function as an outer code. The CRC encoder results in the corresponding outer code and appends CRC bits to the end of the input message such that $w_1^{\mathcal{N}} = u_1^{\mathcal{M}+K+C}$. Subsequently, the generated outer code is encoded as inner code by the polar encoder function that produces PC-CRC-concatenated polar codeword ($x_1^{\mathcal{N}}$) through (1).

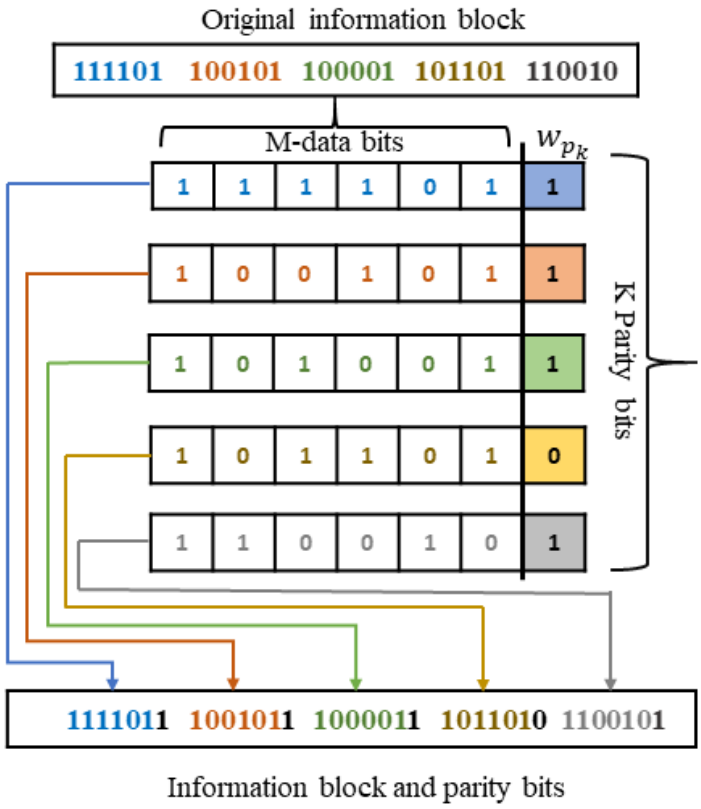

Figure 4. Partitioning of a message vector into equal size segments.

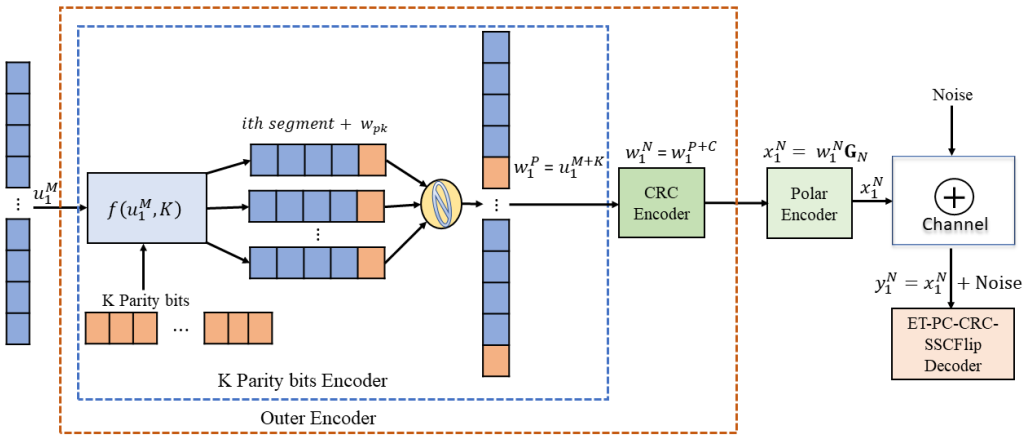

Figure 5. Schematic view of a triple-layer concatenated PC-CRC polar coding scheme.

3.3. Parity-Check-CRC-Aided SSCFlip Decoding

The error correction performance of SCFlip is not satisfactory at a low E_b/N_0, as it requires enormous additional SC decoding attempts, which results in high decoding

complexity. Consequently, we present a parity-check-CRC-concatenated SSCFlip (PC-CRC-SSCFlip) to improve the error correction performance of SCFlip and reduce the decoding complexity and latency. The proposed PC-CRC-SSCFlip utilizes the SSC local decoder and prunes a sub-tree of the Rate-0 node by sending a pre-set resultant vector to its parent node which reduces the decoding latency. The difference between the PC-CRC-SSCFlip decoder (described in this study) and other SCFlip decoders is that each message vector is subdivided into segments. While performing the SSCFlip decoding, a parity bit is computed for each segment by parity function following (7) in parallel and the SSCFlip decoding process is terminated once an error is detected by a parity bit rather than decoding the entire message. In the PC-CRC-SSCFlip decoder, the parity-check vector assists the SSCFlip decoder in intelligently identifying unsuccessful decoding attempts at the early stage and terminates the corresponding decoding process. The distributed parity checks also help the SCL decoder to prune the error paths promptly [8]. The proposed PC-CRC-SSCFlip decoder terminates the SSC decoding process at an early stage once an error is detected by any parity bit. It intelligently identifies the incorrectly estimated bit and corrects it, thus, it can reduce the decoding complexity and latency. A detailed description of the proposed technique is given in Algorithms 1–4, with the following main steps.

1. Initialize the local and global parameters such as the loop control variable (j), information set indices ($\mathcal{A}$), parity bit indices ($\mathcal{P}$), the maximum number of flipping attempts ($\mathcal{T}$), and the number of flips (α).
2. Algorithm 1 begins decoding with standard SC decoding and obtains the estimated information bits $\left(\{\hat{u}_i\}_{i\in\mathcal{A}}\right)$ in line 2. Verify the estimated message $\left(\{\hat{u}_i\}_{i\in\mathcal{A}}\right)$ through the CRC procedure and check the remainder (r) in lines 3 and 4, respectively. If r is zero, break and return $\left(\{\hat{u}_i\}_{i\in\mathcal{A}}\right)$. Otherwise, call Algorithm 2 **(Calculate_Parity(.))** in line 7.
3. **(Calculate_Parity(.))** initializes the local parameters and calculates the parity bits w_{pk} in line 4 by iterating through the parity-check vector K. Compare the computed parity w_{pk} with the corresponding u_{pk} in line 5 and set the $parity\_flag$ value. Return the index $\{i\}_{i=p_k}$ and $parity\_flag$ value to the calling script.
4. Algorithm 1 decides the error-prone bit in line 10 and calls Algorithm 3 **(ET_SSCFlip(.))** in line 11 to flip the error-prone bit. The **ET_SSCFlip(.)** flips and computes the estimated information bit in line 4. The function $h\left(L(\hat{y}_i, i, e\right)$ in Algorithm 3 is used to decide the hard decision through (8).
5. The **ET_SSCFlip(.)** calls Algorithm 2 in line 6 to verify the parity bit. It checks the flag value in line 7 and terminates early once the flag value is zero, and returns to calling script.
6. Algorithm 1 checks the returned flag value in line 12. If the flag value is true in line 12, it passes estimated $\left(\{\hat{u}_i\}_{i\in\mathcal{A}}\right)$ through the CRC procedure and check r in line 14 to verify the estimated message. If r is zero, break. Otherwise, perform the next flipping attempt in lines 9–18.
7. Algorithm 4 is used to flip two erroneous decisions. The flipping mechanism for the first incorrectly estimated bit follows the steps of Algorithm 1 in lines 8–17. It further performs the nested i-th ($2 \le i \le \alpha$) flip in the new trajectory obtained from the previous flips in lines 20–23.

$$h\left(L(\hat{y}_i), i, e\right) = \begin{cases} 0 & i \in \mathcal{A}^c, \\ 1 - \frac{1-sign\left(L(\hat{y}_i)\right)}{2} & i \in e, \\ \frac{1-sign\left(L(\hat{y}_i)\right)}{2} & i \in \mathcal{A}. \end{cases} \tag{8}$$

where e denotes the error-prone bit.

Algorithm 1 Single bit flipping PC-CRC-SSCFlip Algorithm

Input: $llr_1^{\mathcal{N}}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{T}$, $\alpha = 1$
Output: $\{\hat{u}_i\}_{i\in\mathcal{A}}$

Initialize system local and global parameters
$(\{\hat{u}_i\}_{i\in\mathcal{A}}, \{L(\hat{y}_i)\}_{i\in\mathcal{A}}) \leftarrow \mathrm{SC}(llr_1^N, \mathcal{A})$;
$r \leftarrow \mathrm{CRC}(\{\hat{u}_i\}_{i\in\mathcal{A}})$
if $(r == 0)$ **then**
    break;
else
    Calculate_Parity$(\{\hat{u}_i\}_{i\in\mathcal{A}}, \mathcal{A}, \mathcal{P})$
    $L_{Pflip} \leftarrow \{\mathrm{FL}\}_{i<p_k}$
    for $j = 1, 2, \ldots, \mathcal{T}$ **do**
        e $\leftarrow L_{Pflip}(j)$
        ET_SSCFlip$(llr_1^{\mathcal{N}}, \mathcal{A}, \mathcal{P}, e)$
        if $flag == 1$ **then**
            $r \leftarrow \mathrm{CRC}(\{\hat{u}_i\}_{i\in\mathcal{A}})$
            if $(r == 0)$ **then**
                break;
            end if
        end if
    end for
end if
Return $\{\hat{u}_i\}_{i\in\mathcal{A}}$

Algorithm 2 Calculate_Parity$(\{\hat{u}_i\}_{i\in\mathcal{A}}, \mathcal{A}, \mathcal{P})$

Initialization:
$K \leftarrow$ length of $\mathcal{P}$
for $k = 1, \cdots, K$ **do**

$$w_{p_k} = \left(\sum_{i\in\mathcal{A}, i\neq p_k, i<p_k} u_i \right), \ \forall\ p_k \in \mathcal{P}\ \&\ \mathcal{P} \subset \mathcal{A}.$$

    if $(w_{p_k} == u_{pk})$ **then**
        continue;
        $flag \leftarrow 1$
    else
        $flag \leftarrow 0$
        break;
    end if
end for
Return $\{i\}_{i=p_k}$, $flag$

Algorithm 3 ET_SSCFlip($llr_1^N, \mathcal{A}, \mathcal{P}, e$)

Initialization:
for $i = 1, \cdots, \mathcal{N}$ **do**
  $L(\hat{y}_i) = log(\frac{Pr(u_i=0|Y,\hat{u}_1^{i-1})}{Pr(u_i=1|Y,\hat{u}_1^{i-1})})$.
  $\hat{u}_i \leftarrow h(L(\hat{y}_i), i, e)$
  if ($i == p_k$) **then**
    Calculate_Parity$(\{\hat{u}_i\}, \mathcal{A}, \mathcal{K})$ such that $i \in \mathcal{A}, i \leq p_k$
    if $flag == 0$ **then**
      break;
    else
      continue;
    end if
  end if
end for
Return $\{\hat{u}_i\}_{i\in\mathcal{A}}$, $flag$

Algorithm 4 Two bits flipping PC-CRC-SSCFlip Algorithm

Input: $llr_1^N, \mathcal{A}, \mathcal{P}, \alpha = 2$
Output: $\{\hat{u}_i\}_{i\in\mathcal{A}}$
Initialize system local and global parameters
$(\{\hat{u}_i\}_{i\in\mathcal{A}}, \{L(\hat{y}_i)\}_{i\in\mathcal{A}}) \leftarrow SC(llr_1^N, \mathcal{A})$;
$r \leftarrow CRC(\{\hat{u}_i\}_{i\in\mathcal{A}})$
if ($r == 0$) **then**
  break;
else
  Calculate_Parity$(\{\hat{u}_i\}_{i\in\mathcal{A}}, \mathcal{A}, \mathcal{P})$
  $L_{pflip}^1 \leftarrow \{\text{FL}\}_{i<p_k}$
  for $j = 1, 2, \ldots, \mathcal{T}^{(2,1)}$ **do**
    $e_1 \leftarrow L_{pflip}^1(j)$
    ET_SSCFlip$(llr_1^{\mathcal{N}}, \mathcal{A}, \mathcal{P}, e)$
    if $flag == 1$ **then**
      $r \leftarrow \text{CRC}(\{\hat{u}_i\}_{i\in\mathcal{A}})$
      if ($r == 0$) **then**
        Return $\{\hat{u}_i\}_{i\in\mathcal{A}}$
      end if
    end if
    if ($flag \neq 1 \,||\, r \neq 0$) **then**
      $L_{Pflip2} \leftarrow \{\text{FL}\}_{i<p_k}$
      for $m = 1, 2, \ldots, \mathcal{T}^{(2,2)}$ **do**
        $e_2 \leftarrow L_{PFlip2}(m)$ ▷ such that $e_1 \neq e_2$
        ET_SSCFlip$(llr_1^{\mathcal{N}}, \mathcal{A}, \mathcal{P}, \{e_1, e_2\})$
      end for
    end if
  end for
end if
Return $\{\hat{u}_i\}_{i\in\mathcal{A}}$

4. Simulation Results

In this section, we discuss the performance of the proposed PC-CRC-SSCFlip decoder and compare it with the conventional SC, CA-SCL [7], SCFlip [9], improved SCFlip [10], D-SCFlip [20], SC-Fano [31], and PC-SCFlip [24] decoders. The polar codes are constructed according to the Gaussian approximation (GA) [32] with SNR = 2.5 dB and simulations

are conducted under binary phase-shift keying (BPSK) modulation over an additive white Gaussian noise (AWGN) channel for the code length $P(1024, 512)$ with 24 check bits. For a fair comparison, we consider 24 check bits for each decoding algorithm. The proposed algorithm and work in [24] consists of 24 check bits, including 8 parity bits and 16 bits CRC of generator polynomial $g(x) = x^{16} + x^{15} + x^2 + 1$, while the other decoding algorithms consist of 24 bits CRC of generator polynomial $g(x) = x^{24} + x^{23} + x^6 + x^5 + x^1 + 1$. In the proposed work, all the check bits are also configured to be non-frozen bits. Therefore, the actual code rate is equal to $\mathcal{R} = \frac{(\mathcal{K}-\text{check bits})}{\mathcal{N}}$. Accordingly, we evaluate the proposed PC-CRC-SSCFlip decoder in terms of the complexity and error correction performance and compare it with other SC flip decoding algorithms using the same code rate.

4.1. Early Stopping Performance and Decoding Latency

The fascinating factor of the proposed PC-CRC-SSCFlip is that it subdivides each message vector into segments on which SSCFlip is run, and computes the parity bit for each segment rather than decoding the entire message. The PC-CRC-SSCFlip stops early the corresponding decoding process once the parity bit identifies the error.

We record the event gamma (γ) to evaluate the performance of distributed parity checks and demonstrate, in Table 1 for the code-length $P(512, 256)$ with 8-bits parity check concatenated code, that event γ is a percentage of early stopping for the different parity bit. The value of γ for the first parity bit remains steady (i.e., approximately 40%) at the E_b/N_0 ranging from 0.2 to 0.6, but it declines as E_b/N_0 rises. Furthermore, the aggregate value of the γ for the first four bits is above 60% at all E_b/N_0 points. This implies that 60% of the decoding process is terminated early for a segment size less than 50% of the message vector. Consequently, the output of event γ summarized in Table 1 proves that the proposed PC-CRC-SSCFlip decoder avoids unnecessary decoding steps at the early stage and reduces the average decoding latency of the SCFlip decoder.

Table 1. Summary of event γ the PC-CRC-SSCFlip decoder with $\mathcal{N} = 512$, $\mathcal{K} = 256$, $\mathcal{T} = 10$, $\alpha = 1$ and $K = 8$ for 10^7 blocks.

E_b/N_0 (dB)	K Parity Bits							
	1*st*	**2*nd***	**3*rd***	**4*th***	**5*th***	**6*th***	**7*th***	**8*th***
0.2	43%	18%	16%	13%	5%	1%	1%	3%
0.4	43%	16%	14%	12%	6%	2%	2%	5%
0.6	41%	15%	15%	10%	9%	4%	1%	5%
0.8	35%	12%	14%	17%	10%	4%	1%	7%
1	35%	18%	11%	13%	12%	4%	1%	6%
1.2	30%	20%	16%	13%	10%	5%	2%	4%
1.4	28%	17%	13%	16%	12%	7%	1%	6%
1.6	24%	14%	15%	16%	10%	9%	1%	11%
1.8	23%	14%	11%	17%	15%	8%	2%	10%
2	20%	15%	9%	17%	14%	10%	2%	13%

The work in [24] uses a conventional SC decoder as an underlying decoder, and thus the decoding latency for the block-length $\mathcal{N}$ is $2\mathcal{N} - 1$. The non-leaf Rate-1 node v in [24] uses a one clock cycle to compute the LLR a_{vl} for the left child and then waits a particular time T_l until it receives the hard decisions b_{vl} from the left descendent nodes. Similarly, it uses one clock cycle to compute the LLR a_{vr} for the right child and then again waits for another particular time T_r to receive the hard decisions b_{vr} from the right descendent nodes [30]. On the contrary, the proposed work uses SSC decoder as an underlying decoder,

where Rate-1 nodes are instantaneously decoded, and thus it saves $T_l + T_r + 2$ clock cycles. Consequently, it avoids the unnecessary decoding steps of Rate-0 nodes.

In addition, Table 2 shows the decoding clock cycle comparison of the proposed PC-CRC-SSCFlip decoder with the standard SCFlip, and improved SCFlip for $p(1024, 512)$ with 24 check bits. It can be seen that PC-CRC-SSCFlip has low decoding clock cycles due to its early termination property and decreases when the E_b/N_0 increases.

Table 2. Decoding clock cycle comparison of the proposed PC-CRC-SCFlip decoder with SCFlip and the improved SCFlip for $P(1024, 512)$ with 24 check bits.

Decoding Algorithms	E_b/N_0 (dB)					
	1	**1.2**	**1.4**	**1.6**	**1.8**	**2**
SCFlip	46,057	33,160	17395	11,256	9210	6141
Improved SCFlip	27,634	23,540	15,557	10,849	8188	5526
PC-CRC-SCFlip	9211	8410	5607	4005	3043	2082

4.2. Average Computational Complexity

Figure 6 shows the average computational complexity of the PC-CRC-SSCFlip decoder compared with SC, CA-SCL [7], SCFlip [9], improved SCFlip [10], D-SCFlip [20], SC-Fano [31], and PC-SCFlip [24] decoders for $P(1024, 512)$ with 24 check bits. The figure shows that the average computational complexity of the proposed technique is much smaller than SCFlip and identical to the PC-SCFlip decoder, even at low E_b/N_0. However, since the synthetic channels become more reliable at high E_b/N_0 regions, therefore, the average computational complexities converge to the conventional SC decoding complexity, i.e., $O(\mathcal{N} \log \mathcal{N})$. Additionally, the computational complexity of the SC-Fano decoder increases when the value of Δ decreases as this tends to search for more nodes on the code tree. Therefore, SC-Fano with a small Δ value has a higher computational complexity compared with the proposed PC-CRC-SCFlip decoder.

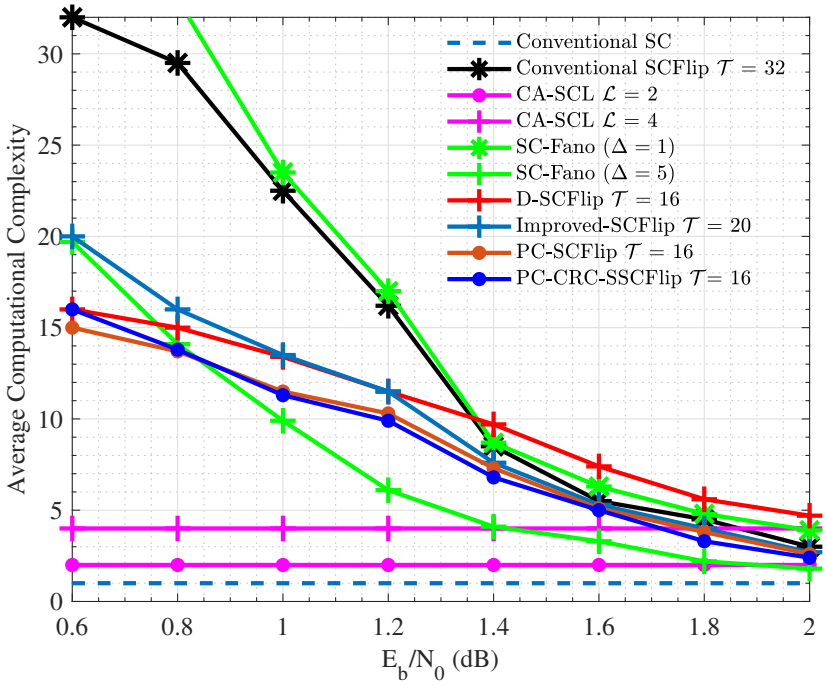

Figure 6. Average computational complexity comparison of the PC-CRC-SSCFlip decoder with different other SC flip decoder for $P(1024, 512)$ with 24 check bits.

Table 3 further compares the proposed PC-CRC-SSCFlip average computational complexity in terms of LLRs computation with other SC flip decoders at a target BLER = 10^{-3}. It can be observed that the proposed method has a smaller average computational complexity compared with other SC flip decoders. The proposed method terminates the decoding process and prevents the computation of unnecessary LLRs whenever a segment does not verify the parity check, leading to computational complexity saving. Moreover, it uses the SSC decoder as an underlying decoder, which does not activate the Rate-0 node. As a result, the number of LLRs computation is reduced.

Table 3. Average LLRs computation comparison of the proposed method with different SC flip decoders for (1024, 512) polar codes with 24 check bits.

Decoding Methods		**SC-Flip [9]**	**SC-Fano (Δ = 1) [31]**	**SC-Fano (Δ = 5) [31]**	**Improved SCFlip [10]**	**D-SCFlip [20]**	**PC-SCFlip [24]**	**PC-CRC-SSCFlip**
Required E_b/N_0 * (dB)		2.7	2.5	2.9	2.6	2.6	2.6	2.6
Complexity ($\times 10^4$)	LLR compute.	3.072	4.096	2.048	2.8672	3.072	2.015	1.766

* denotes the required E_b/N_0 to achieve the target BLER of 10^{-3}.

4.3. Block Error Rate (BLER) Performance

The BLER of the proposed PC-CRC-SSCFlip decoder is compared with the conventional SC, CA-SCL [7], SCFlip [9], improved SCFlip [10], D-SCFlip [20], SC-Fano [31], and PC-SCFlip [24] decoder for $P(1024, 512)$ with 24 check bits. However, the CA-SCL decoder is an efficient algorithm because it significantly improves the BLER performance of SC decoding at the cost of high computational complexity and is considered as a benchmark in the decoding of the polar codes [33]. Moreover, the OA-SCFlip decoder serves as a lower bound for evaluating the BLER performance [10]. Consequently, CA-SCL and OA-SCFlip decoders are considered a baseline for evaluating the BLER performance of the PC-CRC-SCFlip decoder.

The BLER performance of the PC-CRC-SSCFlip is demonstrated in Figure 7. It compares the BLER of the PC-CRC-SSCFlip decoder with SC and SCFlip, and lowers the bound corresponding to the OA-SCFlip. It can be seen that the proposed algorithm approaches the OA-SCFlip-1 at $\mathcal{T} = 16$, while the conventional SCFlip requires 32 additional SC decoding attempts to reach the lower bound of SCFlip-1. For the nested flips with $\alpha = 2$, we set $\mathcal{T}^{2,1} = 16$ to flip the initial erroneous bit and $\mathcal{T}^{2,2} = 5$ for the second erroneous bit in the new decoding trajectory determined by the previous flip. Simulation results show that the proposed algorithm gains over SC and SCFlip while approaching the lower bound of the OA-SCFlip-2. For instance, at a fixed BLER = 10^{-3}, it has up to a 0.7 dB and 0.3 dB gains over SC and SCFlip ($\mathcal{T} = 32$), respectively.

Figure 8 further compares the BLER performance of the PC-CRC-SSFLip decoder for $\alpha = \{1, 2\}$ with those of the CA-SCL and improved-SCFlip decoders. It can be observed that the BLER performance of the proposed algorithm with $\mathcal{T}^{2,1} = 16$ and $\mathcal{T}^{2,2} = 5$ approaches CA-SCL ($\mathcal{L} = 4$) and improved-SCFlip ($\mathcal{T}^{2,1} = 20$ and $\mathcal{T}^{2,2} = 5$) decoders. Additionally, it has a performance gain of 0.35dB at a fixed BLER = 10^{-3} compared with CA-SCL ($\mathcal{L} = 2$).

In addition, Figure 9 compares the BLER performance of the proposed PC-CRC-SCFlip with those of SC-Fano, D-SCFlip, and PC-SCFlip for the $P(1024, 512)$ with 24 check bits. The simulation results show that there are performance gains of 0.3 dB and 0.6 dB at a fixed BLER = 10^{-3} compared with the D-SCFlip and SC-Fano ($\Delta = 5$) decoders, respectively. Moreover, the proposed method with $\alpha = 2$ achieves a 0.3 dB performance gain at a fixed BLER = 10^{-3} and is identical compared with PC-SCFlip ($\alpha = 1$) and PC-SCFlip ($\alpha = 2$), respectively. It can be seen that the error correction performance of the SC-Fano depends on the value of parameter Δ and can be improved by choosing a smaller value of Δ. However, a smaller Δ value increases the computational complexity of the SC-Fano decoder,

as illustrated in Figure 6. Consequently, the proposed technique improves error correction performance with low computational complexity and latency, which is also necessary in critical applications such as healthcare [34] and smart grid [35–40].

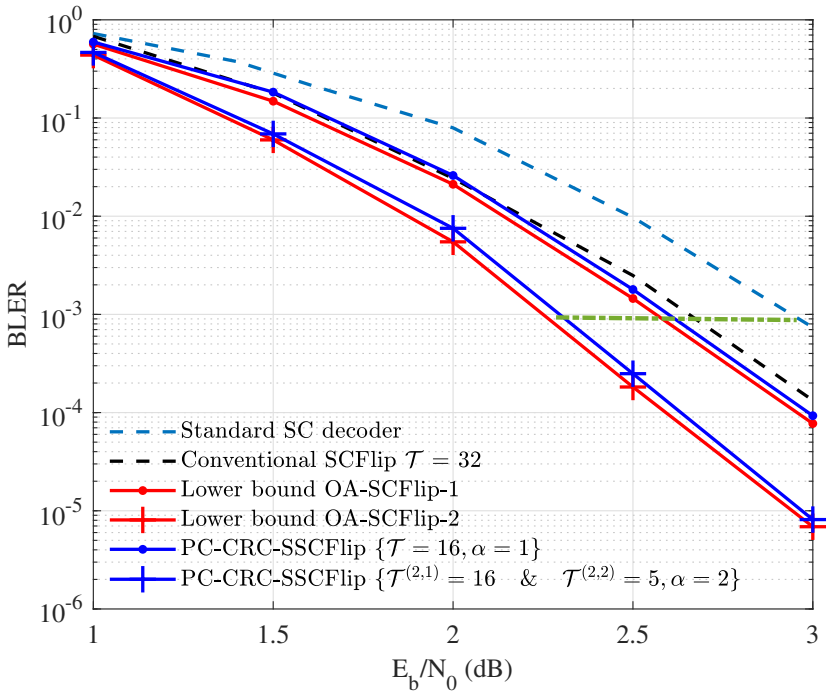

Figure 7. BLER performance comparison of the PC-CRC-SSCFlip decoder with SC, SCFlip, and OA-SCFlip decoders for $P(1024, 512)$ with 24 check bits.

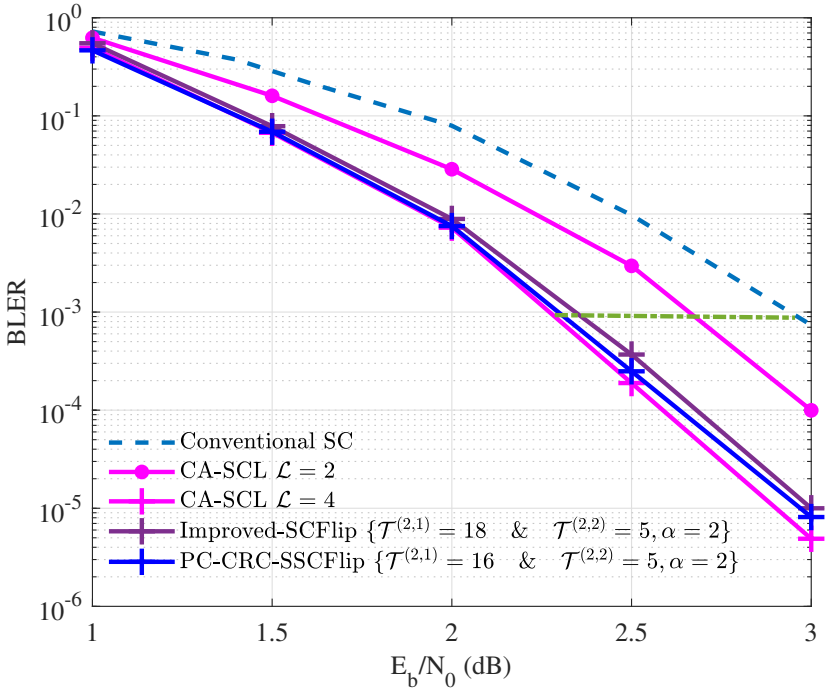

Figure 8. BLER performance comparison of the PC-CRC-SSCFlip decoder with SCFlip, improved-SCFlip, and CA-SCL decoders for $P(1024, 512)$ with 24 check bits.

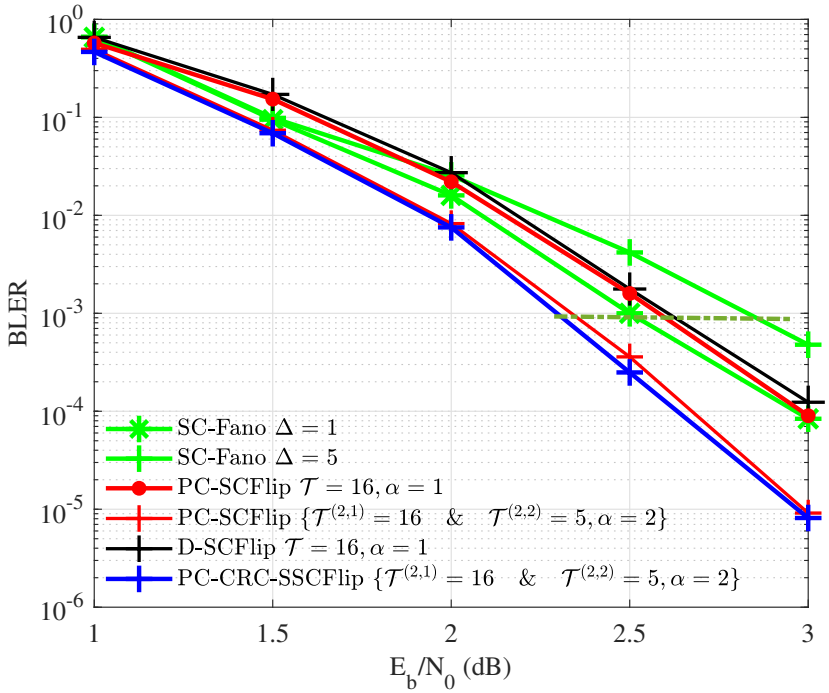

Figure 9. BLER performance of the PC-CRC-SSCFlip decoder compared with SC-Fano, D-SCFlip, and PC-SCFlip decoders for $P(1024, 512)$ with 24 check bits.

5. Conclusions

In this paper, we propose an early termination parity-check-CRC concatenated polar coding scheme for simplified successive cancellation flip decoder (PC-CRC-SSCFlip) for polar codes. The proposed PC-CRC-SSCFlip employs the distributed parity checks to detect an error at an early stage. It stops the corresponding decoding promptly once an error is detected by any parity bit. The proposed PC-CRC-SSCFlip also dynamically constructs an error-prone flipping list by utilizing the distributed parity checks and narrows down the search space for the identification of an error-prone bit. Moreover, it introduces the nest flipping concept into the SSCFlip decoder, which flips more than one erroneous bit in the decoding trajectory resulting from the previous flip. Simulation results show that the proposed technique minimizes the decoding complexity and latency on average, even at low-to-medium SNRs. For instance, 60% of unsuccessful additional SC decoding attempts terminate early rather than decode the entire block. The results also reveal that the BLER performance of the proposed technique is superior to the conventional SC and SCFlip decoder.

Author Contributions: Q.J.: Development of idea and conceptualization, methodology, mathematical modeling, algorithm development, simulations coding, writing—original draft, review, and editing. S.H.: technical support, visualizations, checking and validation of simulations, writing—review and edit the draft. M.F.: Meeting and discussion. Z.P.: Supervision and technical support in the conceptualization, review, and feedback in the draft, checking and validation of simulation, final revision, and approval, funding accusation. N.L.: Support and discussion in the draft preparation. X.Y.: Draft checking and feedback, final validation checking and approval. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Key Research and Development Project under Grant 2018YFB1802402.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Arikan, E. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073. [CrossRef]
2. Balatsoukas-Stimming, A.; Giard, P.; Burg, A. Comparison of Polar Decoders with Existing Low-Density Parity-Check and Turbo Decoders. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
3. Niu, K.; Chen, K.; Lin, J.; Zhang, Q.T. Polar codes: Primary concepts and practical decoding algorithms. *IEEE Commun. Mag.* **2014**, *52*, 192–203. [CrossRef]
4. Tal, I.; Vardy, A. List decoding of polar codes. In Proceedings of the 2011 IEEE International Symposium on Information Theory Proceedings, St. Petersburg, Russia, 31 July–5 August 2011; pp. 1–5.
5. Niu, K.; Chen, K. Stack decoding of polar codes. *Electron. Lett.* **2012**, *48*, 695–697. [CrossRef]
6. Niu, K.; Chen, K. CRC-Aided Decoding of Polar Codes. *IEEE Commun. Lett.* **2012**, *16*, 1668–1671. [CrossRef]
7. Tal, I.; Vardy, A. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2213–2226. [CrossRef]
8. Wang, T.; Qu, D.; Jiang, T. Parity-Check-Concatenated Polar Codes. *IEEE Commun. Lett.* **2016**, *20*, 2342–2345. [CrossRef]
9. Afisiadis, O.; Balatsoukas-Stimming, A.; Burg, A. A low-complexity improved successive cancellation decoder for polar codes. In Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2–5 November 2014; pp. 2116–2120.
10. Chandesris, L.; Savin, V.; Declercq, D. An Improved SCFlip Decoder for Polar Codes. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6. [CrossRef]
11. Zhang, Z.; Qin, K.; Zhang, L.; Zhang, H.; Chen, G.T. Progressive Bit-Flipping Decoding of Polar Codes over Layered Critical Sets. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
12. Sarkis, G.; Giard, P.; Vardy, A.; Thibeault, C.; Gross, W.J. Fast Polar Decoders: Algorithm and Implementation. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 946–957. [CrossRef]
13. Giard, P.; Burg, A. Fast-SSC-flip decoding of polar codes. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, Spain, 15–18 April 2018; pp. 73–77. [CrossRef]
14. Zhou, Y.; Lin, J.; Wang, Z. A New Fast-SSC-Flip Decoding of Polar Codes. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]
15. Zhou, Y.; Lin, J.; Wang, Z. Improved Fast-SSC-Flip Decoding of Polar Codes. *IEEE Commun. Lett.* **2019**, *23*, 950–953. [CrossRef]
16. Hanif, M.; Ardakani, M. Fast Successive-Cancellation Decoding of Polar Codes: Identification and Decoding of New Nodes. *IEEE Commun. Lett.* **2017**, *21*, 2360–2363. [CrossRef]
17. Ercan, F.; Condo, C.; Hashemi, S.A.; Gross, W.J. Partitioned Successive-Cancellation Flip Decoding of Polar Codes. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
18. Wang, X.; Wang, T.; Li, J.; Zhang, Y. Improved Multiple Bit-Flipping Fast-SSC Decoding of Polar Codes. *IEEE Access* **2020**, *8*, 27851–27860. [CrossRef]
19. Condo, C.; Ercan, F.; Gross, W.J. Improved successive cancellation flip decoding of polar codes based on error distribution. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, Spain, 15–18 April 2018; pp. 19–24. [CrossRef]
20. Chandesris, L.; Savin, V.; Declercq, D. Dynamic-SCFlip Decoding of Polar Codes. *IEEE Trans. Commun.* **2018**, *66*, 2333–2345. [CrossRef]
21. Ercan, F.; Condo, C.; Gross, W.J. Improved Bit-Flipping Algorithm for Successive Cancellation Decoding of Polar Codes. *IEEE Trans. Commun.* **2019**, *67*, 61–72. [CrossRef]
22. Guo, R.; Chen, K.; Liu, H. Multi-CRC Polar Codes and M-SCFlip-Based Decoding. *IEEE Access* **2019**, *7*, 98366–98373. [CrossRef]
23. Ercan, F.; Tonnellier, T.; Doan, N.; Gross, W.J. Simplified Dynamic SC-Flip Polar Decoding. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 1733–1737. [CrossRef]
24. Dai, B.; Gao, C.; Yan, Z.; Liu, R. Parity Check Aided SC-Flip Decoding Algorithms for Polar Codes. *IEEE Trans. Veh. Technol.* **2021**, *70*, 10359–10368. [CrossRef]
25. Yang, D.; Yang, K. Error-Aware SCFlip Decoding of Polar Codes. *IEEE Access* **2020**, *8*, 163758–163768. [CrossRef]
26. Jan, Q.; Hussain, S.; Zechen, L.; Zhiwen, P.; Nan, L.; Xiaohu, Y. Improved Partitioned Fast-SSC-Flip Decoding for Polar Codes. In Proceedings of the 2022 7th International Conference on Computer and Communication Systems (ICCCS), Wuhan, China, 22–25 April 2022; pp. 382–386. [CrossRef]
27. Yu, Y.; Pan, Z.; Liu, N.; You, X. Belief Propagation Bit-Flip Decoder for Polar Codes. *IEEE Access* **2019**, *7*, 10937–10946. [CrossRef]
28. Yang, Y.; Yin, C.; Qasim, J.; Hu, Y.; Pan, Z.; Liu, N.; You, X. Noise-Aided Belief Propagation List Bit-Flip Decoder for Polar Codes. In Proceedings of the 2020 International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, China, 21–23 October 2020; pp. 807–810. [CrossRef]
29. Jan, Q.; Hussain, S.; Furqan, M.; Pan, Z.; Liu, N.; You, X. A Novel Flip-List-Enabled Belief Propagation Decoder for Polar Codes. *Electronics* **2021**, *10*, 2302. [CrossRef]

30. Alamdar-Yazdi, A.; Kschischang, F.R. A Simplified Successive-Cancellation Decoder for Polar Codes. *IEEE Commun. Lett.* **2011**, *15*, 1378–1380. [CrossRef]
31. Jeong, M.O.; Hong, S.N. SC-Fano Decoding of Polar Codes. *IEEE Access* **2019**, *7*, 81682–81690. [CrossRef]
32. Trifonov, P. Efficient Design and Decoding of Polar Codes. *IEEE Trans. Commun.* **2012**, *60*, 3221–3227. [CrossRef]
33. 3GPP Groups. Final Report of 3GPP TSG RAN WG1 AH1 NR v1.0.0. In Proceedings of the 3GPP TSG RAN WG1 Meeting 88, R1-1701553, Athens, Greece, 13–17 February 2017.
34. Shukla, S.; Thakur, S.; Hussain, S.; Breslin, J.G.; Jameel, S.M. Identification and authentication in healthcare internet-of-things using integrated fog computing based blockchain model. *Internet Things* **2021**, *15*, 100422. [CrossRef]
35. Hussain, S.; Thakur, S.; Shukla, S.; Breslin, J.G.; Jan, Q.; Khan, F.; Kim, Y.S. A two-layer decentralized charging approach for residential electric vehicles based on fuzzy data fusion. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 7391–7405. [CrossRef]
36. Hussain, S.; Kim, Y.S.; Thakur, S.; Breslin, J.G. Optimization of Waiting Time for Electric Vehicles Using a Fuzzy Inference System. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 15396–15407. [CrossRef]
37. Hussain, S.; Thakur, S.; Shukla, S.; Breslin, J.G.; Jan, Q.; Khan, F.; Ahmad, I.; Marzband, M.; Madden, M.G. A heuristic charging cost optimization algorithm for residential charging of electric vehicles. *Energies* **2022**, *15*, 1304. [CrossRef]
38. Hussain, S.; Ahmed, M.A.; Kim, Y.C. Efficient power management algorithm based on fuzzy logic inference for electric vehicles parking lot. *IEEE Access* **2019**, *7*, 65467–65485. [CrossRef]
39. Hussain, S.; Ahmed, M.A.; Lee, K.B.; Kim, Y.C. Fuzzy logic weight based charging scheme for optimal distribution of charging power among electric vehicles in a parking lot. *Energies* **2020**, *13*, 3119. [CrossRef]
40. Hussain, S.; Lee, K.B.; Ahmed, M.A.; Hayes, B.; Kim, Y.C. Two-stage fuzzy logic inference algorithm for maximizing the quality of performance under the operational constraints of power grid in electric vehicle parking lots. *Energies* **2020**, *13*, 4634. [CrossRef]