IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 24, 2019, accepted August 31, 2019, date of publication September 11, 2019, date of current version September 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2940525

Low-Latency Adaptive Ordered Statistic Decoding of Polar Codes

KANGJIAN QIN, (Member, IEEE), AND ZHAOYANG ZHANG, (Member, IEEE)
College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China
Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking, Zhejiang University, Hangzhou, China

Corresponding author: Zhaoyang Zhang (ning_ming@zju.edu.cn)

This work was supported in part by the China Scholarship Council, in part by the National Key Research and Development Program of China under Grant 2018YFB1801104, and in part by the National Natural Science Foundation of China under Grant 61725104 and Grant 61631003.

ABSTRACT Deploying polar codes in ultra-reliable low-latency communication (URLLC) is of critical importance and is currently receiving tremendous attention in both academia and industry. However, most of the state of the art polar codes decoders like progressive bit-flipping decoder (PBF) and successive cancellation list (SCL) decoder, involve strong data dependencies and suffer from huge decoding delay. This contradicts the low-latency requirement in URLLC. To address such issue, this paper appeals to the parallel computing and proposes an adaptive ordered statistic decoder (OSD). In particular, we first propose a novel codeword searching metric which proves to be hardware-friendly, and an adaptive OSD algorithm is then developed to adaptively rule out the unpromising codewords, thus significantly reducing the latency. Secondly, to further reduce the computational complexity of the proposed algorithm, we decompose the current code sequence into several independent subcodes, and by handling these subcodes with concatenated adaptive OSDs, a good trade-off between decoding latency and complexity can be achieved. Finally, numerical results show that the proposed adaptive OSD outperforms the conventional decoders in terms of block error rate (BLER) and decoding latency.

INDEX TERMS Low latency decoding, ordered statistic decoding, polar codes.

I. INTRODUCTION

Ultra-reliable and low-latency communication (URLLC) is one of the most important scenarios in 5G communications and beyond. Different from the current 4G system, where block error rates (BLERs) are typical around 10^{-2} and time-tolerant retransmission mechanisms like HARQ can be adopted, URLLC requires more stringent BLER (from 10^{-9} to 10^{-5}) which has to be achieved within millisecond order (see Fig. 1) [2]. However, the current coding schemes cannot properly handle the conflict between reliability and latency. As such, new coding methods that meet both requirements become an emerging research trend.

Polar codes [3], as the coding scheme for control channel of enhanced mobile broadband (eMBB) in 5G [4], seem to be a promising candidate for URLLC. With progressive bit-flipping (PBF) decoding [5], superior BLER performance can be achieved with low complexity in high SNR regime. With cyclic redundancy check (CRC) aided successive cancellation list (CA-SCL) decoder [6]–[8], polar codes outperform state-of-the-art turbo and low-density parity-check (LDPC) codes under short block length. Although short block length can reduce the end-to-end (E2E) transmission latency, however, huge decoding latency still prevents existing SC-based decoders from being deployed in URLLC. To solve this, a simplified successive cancellation decoder was first proposed in [9] to simultaneously decode all rate-1 and rate-0 constituent codes. However, the latency gain of such method is highly dependent on the distribution of unfrozen bits. To overcome this limitation, precomputation technique was introduced in [10], [11], where 50% decoding latency was saved at the cost of twice extra memory. Similar trade-off was presented in [12], where lower decoding delay is achieved but higher complexity is required. To preserve the complexity, a stage-reduced SC decoding algorithm and its low-latency architecture were introduced in [13]. Subsequently, a double thresholding algorithm and a split reduction method were proposed to further improve the latency performance of CA-SCL decoders in [14] and [15], respectively. Albeit significantly reducing the decoding latency,

The associate editor coordinating the review of this manuscript and approving it for publication was Yi Fang.

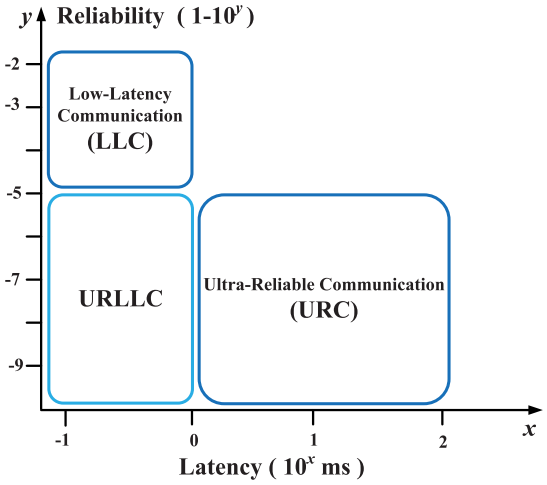


FIGURE 1. URLLC key performance indicators.

the aforementioned SC-based decoders still suffer from data dependency, which means one data bit can only be decoded until all its former bits are derived. In this sense, the decoding latency inevitably increases with the block length.

To avoid such data dependency, ordered statistic decoder (OSD) was proposed. Invoking a different decoding structure, OSD is well suitable for parallel implementation and is able to concurrently estimate all polar codeword bits. This parallel computing nature dramatically reduces the decoding latency. For lower BLER, a level-l OSD was developed [17]. It allows at most l-level reprocessing, yet increasing the decoding delay and complexity. The authors in [18] proposed to use a Gaussian threshold to eliminate the unpromising candidate codewords, thus consuming fewer clock cycles (CCs). However, such threshold is empirical, and it incurs an error floor in high signal-to-noise ratio (SNR) regime, which is definitely unacceptable in URLLC systems.

To address the issues above, an adaptive OSD algorithm is developed in this paper. Specifically, a codeword searching metric is carefully designed. We show that the codeword with minimum searching metric turns out to be the most promising candidate. On this basis, codewords with large metric are considered unpromising and should be eliminated. Based on this searching metric, the adaptive OSD can easily rule out the unpromising candidates throughout the tested codewords and simplify the decoding procedure. For this reason, the decoding latency is greatly decreased. To further reduce the computational complexity, we propose to decompose the current test codeword into several independent subcodes with smaller block length, and decode them with concatenated adaptive OSDs. Due to the reduced codeword dimension, the computational complexity of adaptive OSD is further decreased, and a good trade-off between decoding complexity and latency can be achieved. Numerical results validate that the proposed adaptive OSD decoder outperforms the existing decoders in terms of decoding latency and BLER.

Our main contributions are summarized as follows:

- A novel codeword searching metric is proposed, which proves to minimize the decoding error. Moveover, the proposed metric can be realized with only additive operation, thus is well suitable for hardware implementation.
- An adaptive OSD algorithm, aiming to adaptively ruling out the unpromising candidate codewords, is developed. Due to the reduced number of test codewords, the decoding latency is dramatically decreased.
- Low complexity implementation of adaptive OSD is investigated, which further reduces the complexity of overall decoding process. In addition, a concatenated adaptive OSD is proposed, which achieves a good trade-off between decoding complexity and latency.

The remainder of this paper is organized as follows. In Section II, the preliminaries of polar codes are briefly reviewed. a codeword searching metric and an adaptive OSD algorithm are introduced in Section III. In Section IV, a concatenated adaptive OSD scheme is proposed. Section V gives the relevant numerical results and performance analysis. And finally, Section VI concludes the work.

II. PRELIMINARIES

To better understand the decoding latency, we first briefly review the encoding and decoding procedures of polar codes. Then, SC and SCL decoders are expounded in detail, after which their decoding latencies are also investigated.

A. POLAR CODES

We use a_1^N to denote a sequence $(a_1, a_2, \ldots, a_N)$. For polar codes with block length $N = 2^n$ and kernel $\boldsymbol{F}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, we denote u_1^N as the information sequence, and a polar codeword c_1^N is obtained by $c_1^N = u_1^N \boldsymbol{B}_N \boldsymbol{F}_2^{\otimes n}$, where '$\otimes$' denotes the Kronecker product and $\boldsymbol{B}_N$ is a permutation matrix. A coding rate $R = K/N$ means that a set $\mathcal{A} \subset \{1, 2, \ldots, N\}$ of cardinality K is selected as the information set (see [3]), and thus u_1^N consists of K unfrozen bits $\mathbf{u}_{\mathcal{A}}$ and $N-K$ frozen bits $\mathbf{u}_{\mathcal{A}^c}$ (all frozen bits are assumed to be zero codewords in this paper if not specified). The split-channel is defined as $W_N^{(i)}(y_1^N, u_1^{i-1}|u_i) = \sum_{u_{i+1}^N \in \mathcal{X}^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N|u_1^N)$, and the Bhattacharyya parameter $Z(W_N^{(i)})$ is computed to select the K most reliable split-channels to transmit unfrozen bits. For more details, we refer the reader to [3].

B. SC AND SCL DECODING

The received sequence from a transmitted codeword c_1^N is represented as y_1^N. With these values, the information sequence u_1^N can be estimated using a SC decoder. Since the decoder already knows the frozen bits, it only needs to estimate unfrozen bits using

$$\hat{u}_i = \begin{cases} 0, & L(\hat{u}_i) \geq 0 \\ 1, & L(\hat{u}_i) < 0 \end{cases} \tag{1}$$

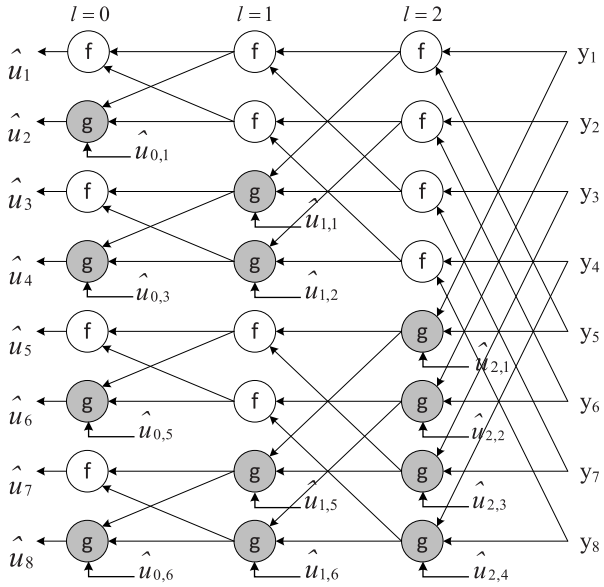


FIGURE 2. A decoding process for polar codes with $N = 8$.

where

$$L(\hat{u}_i) \triangleq \ln \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} | u_i = 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} | u_i = 1)} \tag{2}$$

is the log-likelihood ratio (LLR) corresponding to $\hat{u}_i$. With the butterfly network shown in Fig. 2, we denote the LLR from the underlying channel as $L_{n,i}$, and $L_{0,i} = L(\hat{u}_i)$ is LLR from the bit-channel $W_N^{(i)}$. For other LLR that passes through in Fig. 2, it is denoted by $L_{l,i}$, where l and i correspond to the index of the decoding stage and the row, respectively. Then all these LLRs can be calculated using

$$L_{l,i} = \begin{cases} f(L_{l+1,i}; L_{l+1,i+2^l}) & \text{if } \left\lceil \frac{i}{2^l} \right\rceil \text{ is odd} \\ g(\hat{v}_{l,i-2^l}; L_{l+1,i-2^l}; L_{l+1,i}) & \text{if } \left\lceil \frac{i}{2^l} \right\rceil \text{ is even} \end{cases} \tag{3}$$

where $\hat{v}$ is a modulo-2 partial sum of intermediate decoded bits, $0 \le l < n$, $1 \le i \le N$. Function f and g are defined as:

$$f(\alpha, \beta) \triangleq \ln \left(\frac{e^{\alpha+\beta} + 1}{e^{\alpha} + e^{\beta}} \right), \tag{4}$$

$$g(\alpha, \beta, \hat{v}) \triangleq (-1)^{\hat{v}} \alpha + \beta, \tag{5}$$

respectively.

Note that the value of $\hat{v}$ is required in g function. This introduces strong data dependencies in SC decoding, which further increases latency. The scheduling of SC decoding for polar codes in Fig. 2 is illustrated in Fig. 3. For polar codes with block-length $N = 2^n$, there are n decoding stages. At some stage l, a maximum of 2^l LLR values can be simultaneously obtained. To finish the decoding, a stage indexed by l need to be activated 2^{n-l} times, and each can be done in one CC. As such, the total number of CCs needed for SC decoding is

$$\tau_{SC} = \sum_{l=0}^{n-1} 2^{n-l} = 2N - 2, \tag{6}$$

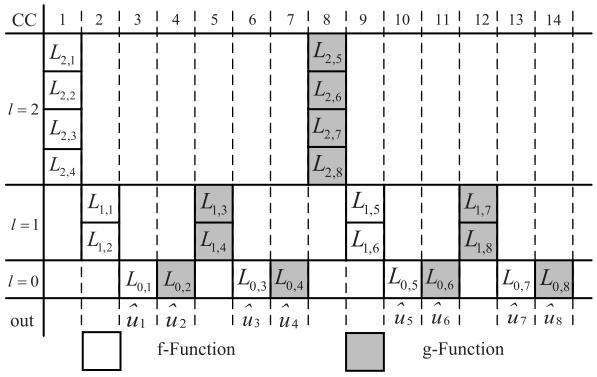


FIGURE 3. The scheduling for a SC decoder with $N = 8$.

which means the decoding latency linearly increases with the block-length in SC decoding.

For SCL decoding, similar logic can be used. However, a SCL decoder with a list size of L inspects both options for the estimate of any unfrozen bit u_i, and a sorting operation is performed to select at most L most promising decoding paths from $2L$ expanded paths [6]. Assuming that all the L candidate paths are decoded in parallel, and the sorting process of $2L$ paths can be done with $2L \log_2(2L)$ comparisons using well studied merge sort algorithm [19]. With sufficient hardware parallelization, the sorting latency is assumed to be one clock cycle in this paper. Then the decoding latency of SCL decoding is approximated by the sum of (6) and the latency for the sorting processes. For polar codes with block-length N and coding rate R, the decoding latency under SCL decoding with a list size of L can be approximated as follows

$$\tau_{SCL} = 2N - 2 + NR - \log_2 L \tag{7}$$

One should note that, data transmitted in URLLC use much shorter block-length to reduce the E2E latency, which leads to a higher coding rate for the same size of information. In this regard, the decoding latency for a SCL decoder is much higher than that of SC decoder.

III. ADAPTIVE ORDERED STATISTIC DECODING

To avoid the data dependency involved in SC decoding, the parallel computing feature of OSD is firstly exploited to simultaneously decode all bits in a codeword. Then, we propose a novel codeword searching metric which efficiently finds the most likely codeword. Finally, an adaptive OSD decoding scheme is developed.

A. OSD AND L-LEVEL OSD

Different from the SC-based decoders whose decoding latencies extend linearly with the block-length, the OSD provides a low-latency implementation of decoding by virtue of its parallel computing for all codeword bits. For a given received sequence y_1^N, we first define a permutation function λ_1 which reorders y_1^N according to their absolute values of LLR. Using the obtained λ_1, we can permute the columns of generator matrix $\mathbf{G}$ to get $\mathbf{G}' = \lambda_1(\mathbf{G})$, where $\mathbf{G}$ is a $K \times N$ submatrix

of $\boldsymbol{B}_N\boldsymbol{F}_2^{\otimes n}$ obtained by removing the rows in $\boldsymbol{B}_N\boldsymbol{F}_2^{\otimes n}$ corresponding to $\mathcal{A}^c$. Next, a new $K \times N$ matrix $\mathbf{G}''$ is constructed from the permuted matrix $\mathbf{G}'$, whose first K columns are the K most reliable independent (MRI) columns of $\mathbf{G}'$, while the rest columns are generated using the rest $N - K$ columns of $\mathbf{G}'$. Note that this operation can also be defined as a permutation function λ_2, i.e., $\mathbf{G}'' = \lambda_2(\mathbf{G}')$, which means $\mathbf{G}''$ can be obtained from $\mathbf{G}$ using one permutation function $\lambda = \lambda_1\lambda_2$ by $\mathbf{G}'' = \lambda(\mathbf{G})$. Finally, the systematic form of $\mathbf{G}''$ is denoted by $\mathbf{G}_s = [\mathbf{I}, \mathbf{P}]$, where $\mathbf{I}$ and $\mathbf{P}$ are identity matrix and check matrix, respectively.

For decoding, the OSD concurrently performs hard decision on the first K components of $\lambda(y_1^N)$ to get a_1^K, and for ease of exposition, a_1^K is referred to as the MRI sequence in this paper. Then if there is no erroneous hard decision in MRI sequence, the final estimated codeword $\hat{c}_1^N$ can be reconstructed in one shot decoding by

$$\hat{c}_1^N = \lambda^{-1}(a_1^K\mathbf{G}_s) \tag{8}$$

However, there are wrong hard decisions due to the channel noise. To achieve lower BLER, a l-level ($1 \le l \le K$) OSD allows at most l level reprocessing. At the i-th ($1 \le i \le l$) level, the decoder flips i bits of a_1^K. Such reprocessing is equivalent to adding an error pattern e_1^K upon a_1^K in binary field. We use $\hat{c}_1^N(e_1^K)$ to represent the candidate codeword corresponding to a test error pattern e_1^K. After all the possible candidate codewords are tested, the decoder selects the codeword $\hat{c}_1^N(e_1^K)$ which has the smallest Euclidean distance with y_1^N as the final output. For notational simplicity, the vector e_1^K in $\hat{c}_1^N(e_1^K)$ is omitted throughout the rest of this paper.

B. DECODING LATENCY OF L-LEVEL OSD

Note that there is no data dependency when decoding bits in a candidate codeword $\hat{c}_1^N$, and all codeword bits can be derived concurrently under the matrix transposition in (8). To obtain the permutation function λ, one needs first to sort the original received sequence y_1^N according to their absolute LLR values, and this process can be done with $N\log_2 N$ comparisons whose latency is assumed to be one clock cycle under adequate parallelization. Then, the overall process of transforming $\mathbf{G}'$ to $\mathbf{G}_s$ can be realized by $N \cdot \min(K, N-K)^2$ binary operations with Gaussian elimination. Nevertheless, two levels of parallelism are possible, yielding K steps of at most K independent summations, each summation consisting of N independent binary additions. One should note that such matrix computation can be accomplished with sufficient hardware parallelization so its latency is negligible. Thus the dominant factor of decoding latency for a l-level OSD is the number of error patterns that are reprocessed, which can be approximated by

$$\tau_{OSD_{\text{level}=l}} = \sum_{i=0}^{l}\binom{K}{i} \tag{9}$$

To reduce the number of candidate error patterns, Gaussian threshold was introduced for OSD [18], where only most reliable independent symbols with low confidences are flipped. Binary phase shift keying (BPSK) modulation under memoryless AWGN channel is considered for the ensuing analysis. Then the received symbol can be expressed by $y_i = x_i + n_i$, where $x_i = 1 - 2c_i$, and n_i is a Gaussian random variable with mean zero and variance $\sigma^2 = \frac{N_0}{2}$. On this basis, the probability density function (PDF) of the received symbols, i.e., $\mathcal{N}(1, \sigma)$ and $\mathcal{N}(-1, \sigma)$, can be divided by Gaussian thresholds like $|y_{th}| = \pm(3\sigma - 1)$, into 3 areas as shown in Fig. 4.

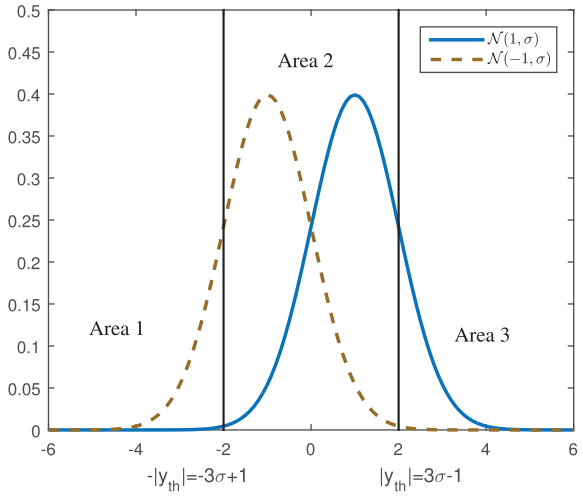


FIGURE 4. Area division using 3σ Gaussian thresholds.

Given a received symbol lying in Area 3, there is a high probability that the transmitted symbol is $+1$ because the probability that the symbol -1 falls into Area 3 is rather small. Similarly, for received symbols in Area 1, they also have a high confidence. In this sense, we only need to consider the error patterns that flip the symbols in Area 2. Apparently, the smaller the absolute value of the threshold is, the less error patterns will be considered. This decreases the decoding latency while increasing the BLER performance. It is shown in [18] that when a threshold of $|y_{th}| = \pm(3\sigma - 1)$ is adopted, this Gaussian threshold based OSD needs only 10% clock cycles of (9) at a SNR of 1.5 dB.

Nevertheless, there are some drawbacks that prevent the above scheme from being deployed in URLLC.

- For the threshold-based OSD, there is a fixed non-zero probability of missing out a true error pattern, which causes an error floor in high SNR region, and this is unbearable in URLLC.
- One has to experimentally determine the Gaussian threshold, and the optimal trade-off between the BLER performance and decoding complexity can hardly be found. (see [18]).
- The above Gaussian threshold is based on real number, however, the received symbols and their corresponding LLRs are quantized into fixed-point values in a practical decoder, and it makes the above Gaussian threshold scheme invalid.

C. A NOVEL CODEWORD SEARCHING METRIC

To avoid the drawbacks of threshold-based OSD, a novel codeword searching metric is firstly proposed. For a given error pattern e_1^K, its associated reconstructed codeword and corresponding BPSK modulated sequence are denoted by $\hat{c}_1^N$ and m_1^N, respectively. Then the Euclidean distance between the received sequence y_1^N and m_1^N can be computed by

$$d(m_1^N, y_1^N) = \sum_{i=1}^{N}(1+y_i^2) - 2\sum_{i=1}^{N} m_i y_i \tag{10}$$

As $\sum_{i=1}^{N}(1+y_i^2)$ is a constant for a certain y_1^N, $\sum_{i=1}^{N} m_i y_i$ is commonly used as a cost function to replace the Euclidean distance, and the codeword which maximizes $\sum_{i=1}^{N} m_i y_i$ is taken as the final estimation.

However, the multiply operations involved in cost function are impractical for hardware realization. By denoting the i-th element in y_1^N and $\hat{c}_1^N$ as y_i and $\hat{c}_i$, respectively, and $h(\cdot)$ denotes the hard decision function, we give the following proposition which sheds light on our main result.

Proposition 1: Given that all codewords have the same *a priori* probability, then finding a codeword that minimizes the decoding error is equivalent to searching a codeword that minimizes the metric in (11).

$$\mathcal{M}\left(\hat{c}_1^N\right) \triangleq \sum_{\substack{1\le i\le N\\ \hat{c}_i\neq h(y_i)}} |y_i| \tag{11}$$

Proof: For whole candidate codewords, the probability of decoding error is minimized by finding a codeword $\hat{c}_1^N$ which maximizes the *a posteriori* probability $P(c_1^N|y_1^N)$ over the codebook $\mathcal{C}$ as follows

$$\hat{c}_1^N = \arg\max_{c_1^N\in\mathcal{C}} P(c_1^N|y_1^N) \tag{12}$$

Since we restrict to memoryless AWGN channel, and all codewords are equiprobable, (12) is equivalent to

$$\hat{c}_1^N = \arg\max_{c_1^N\in\mathcal{C}} \sum_{i=1}^{N} \ln P(y_i|c_i) \tag{13}$$

Note that y_1^N is a determined sequence in one certain transmission, thus $\sum_{i=1}^{N}\ln P(y_i|1)$ is a constant, which means

$$\begin{aligned}\hat{c}_1^N &= \arg\max_{c_1^N\in\mathcal{C}} \sum_{i=1}^{N}\ln P(y_i|c_i) - \sum_{i=1}^{N}\ln P(y_i|1)\\ &= \arg\max_{c_1^N\in\mathcal{C}} \sum_{i=1}^{N}\ln\frac{P(y_i|c_i)}{P(y_i|1)} = \arg\max_{c_1^N\in\mathcal{C}}\sum_{i=1}^{N}(1-c_i)L_i\\ &= \arg\max_{c_1^N\in\mathcal{C}} \sum_{i=1}^{N}\frac{1}{2}(1-2c_i)L_i + \frac{1}{2}\sum_{i=1}^{N}L_i\end{aligned} \tag{14}$$

Note that $L_i = \frac{2y_i}{\sigma^2}$ for AWGN channels, thus $\frac{1}{2}\sum_{i=1}^{N}L_i$ is determined once y_1^N is determined. This leads us to

$$\begin{aligned}\hat{c}_1^N &= \arg\max_{c_1^N\in\mathcal{C}} \sum_{i=1}^{N}\frac{1}{2}(1-2c_i)L_i\\ &= \arg\max_{c_1^N\in\mathcal{C}} \sum_{i=1}^{N}|L_i| - \sum_{\substack{1\le i\le N\\ \hat{c}_i\neq h(y_i)}} 2|L_i|\\ &= \arg\min_{c_1^N\in\mathcal{C}} \sum_{\substack{1\le i\le N\\ \hat{c}_i\neq h(y_i)}} |L_i|\end{aligned} \tag{15}$$

Finally, we have

$$\hat{c}_1^N = \arg\min_{c_1^N\in\mathcal{C}} \sum_{\substack{1\le i\le N\\ \hat{c}_i\neq h(y_i)}} |y_i| \tag{16}$$

which completes the proof. ■

Remarks: To calculate $\mathcal{M}\left(\hat{c}_1^N\right)$, one only needs less than N additions. Since no multiplication is involved, the proposed searching metric can be efficiently implemented in hardware. We further note that, a promising candidate codeword $\hat{c}_1^N$ should have a smaller $\mathcal{M}\left(\hat{c}_1^N\right)$ value according to the *Proposition 1*, which motivates us to skip codewords whose metric values are large. Specially, if current candidate codeword has a larger searching metric value compared to the former tested one, then such candidate codeword is considered to be unpromising and should be ruled out.

D. ADAPTIVE OSD

Capitalize on the above codeword searching metric, an algorithm is developed to adaptively eliminate the unpromising candidate codeword, so as to avoid the drawbacks caused by Gaussian threshold in Section III-B.

Note that given an test error pattern e_1^K, its corresponding candidate codeword is $\hat{c}_1^N(e_1^K)$, which can be computed by adding $e_1^K[\mathbf{I},\mathbf{P}]$ to $\hat{c}_1^N(0_1^K)$. We refer to $\tilde{c}_1^N = e_1^K[\mathbf{I},\mathbf{P}]$ as the codeword error pattern. On this basis, the searching metric can be decomposed into two parts

$$\begin{aligned}\mathcal{M}\left(\hat{c}_1^N\right) &= \sum_{\substack{1\le i\le K\\ e_i=1}}|\tilde{y}_i| + \sum_{\substack{K+1\le i\le N\\ \tilde{c}_i=1}}|\tilde{y}_i|\\ &= \mathcal{D}(e_1^K) + \mathcal{R}(e_1^K)\end{aligned} \tag{17}$$

where $\tilde{y}_i$ is the i-th element of $\tilde{y}_1^N = \lambda(y_1^N)$, and e_i and $\tilde{c}_i$ is the i-th element of e_1^K and $\tilde{c}_1^N = e_1^K[\mathbf{I},\mathbf{P}]$, respectively. The first part of (17) is referred to as the determined part of searching metric, and is denoted by $\mathcal{D}(e_1^K)$, which can be determined directly according to the current test error pattern e_1^K. While the second part is the redundancy part, denoted by $\mathcal{R}(e_1^K)$, whose value can be calculated according to the redundancy part of codeword error pattern, i.e., $\tilde{c}_{k+1}^N$.

Although the codeword searching metric $\mathcal{M}\left(\hat{c}_1^N\right)$ can be efficiently computed given the reconstructed codeword $\hat{c}_1^N$, however, to reconstruct $\hat{c}_1^N$, its codeword error pattern $\tilde{c}_1^N$ has

to be computed. Note that $\mathcal{D}(e_1^K)$ can be calculated directly using e_1^K, and it is more desirable to get $\mathcal{R}(e_1^K)$ without any reconstruction in terms of hardware implementation. To this end, we propose to estimate $\mathcal{R}(e_1^K)$ using $\mathcal{D}(e_1^K)$ as follows

$$\hat{\mathcal{R}}(e_1^K) = \mathcal{D}(e_1^K)\frac{\sum_{K+1\le i\le N}|\tilde{y}_i|}{\sum_{1\le i\le K}|\tilde{y}_i|} \tag{18}$$

In this regard, the metric associated with the current candidate codeword $\hat{c}_1^N$ can be efficiently estimated using

$$\hat{\mathcal{M}}\left(\hat{c}_1^N\right) = \mathcal{D}(e_1^K) + \hat{\mathcal{R}}(e_1^K) \tag{19}$$

Since a candidate codeword $\hat{c}_1^N$ with smaller $\mathcal{M}(\hat{c}_1^K)$ value is more likely to be the codeword that has been transmitted, whereas the codeword that possesses the largest $\mathcal{M}(\hat{c}_1^K)$ value so far is considered to be unpromising and should be discarded, we discard $\hat{c}_1^N$ (or its corresponding test error pattern e_1^K) if $\hat{\mathcal{M}}(\hat{c}_1^N)$ is greater than the metric value of current most likely codeword.

To implement the adaptive OSD, we use CRC to judge if the current candidate codeword is correct. The algorithm starts by checking the candidate codeword with test error pattern 0_1^K, if it passes the CRC check, then the algorithm returns the current codeword and stops; otherwise, at most $\sum_{i=1}^{l}\binom{K}{i}$ reprocessing is required. The reprocessing procedure checks the remaining test error pattern in a level-by-level manner, i.e., start from patterns with hamming weight 1 to weight l. For a certain level, the error pattern with larger decimal value is checked first. The overall algorithm is summarized as Algorithm 1.

Algorithm 1 Adaptive OSD Decoding

Input: the received sequence y_1^N
Output: the decoded codeword $\hat{c}_1^N$

```
max_level = l, cur_level = 0, e_1^K = 0_1^K
ĉ_1^N ← OSD(y_1^N, e_1^K)
// Initialization
while CRC(ĉ_1^N) = failure and cur_level < l do
    cur_level=cur_level+1
    M̂(ĉ_1^N) = D(e_1^K) + R(e_1^K)
    M*(ĉ_1^N) ← M̂(ĉ_1^N)
    enumerate the {e_1^K | Σ_{i=1}^K e_i = cur_level}
    select an untested e_1^K from {e_1^K} in ascending order
    of Σ_{i=1}^K 2^{K-i} e_i
    calculate M̂(ĉ_1^N) = D(e_1^K) + R̂(e_1^K)
    if M̂(ĉ_1^N) > M*(ĉ_1^N) then
        go to step 8
    else
        ĉ_1^N ← OSD(y_1^N, e_1^K)
        go to step 3
return ĉ_1^N
```

IV. LOW COMPLEXITY IMPLEMENTATION

Although the adaptive OSD is designed to expurgate the unnecessary codeword reprocessing, however, in each reprocessing episode, there are Gaussian eliminations that dominate the complexity. In this section, we first propose to decompose the current code sequence into several independent subcodes, due to the reduced codeword dimension, the implementation complexity of adaptive OSD can be reduced. On this basis, a concatenated adaptive OSD is finally developed, which achieves a good trade-off between decoding latency and complexity.

A. POLAR DECOMPOSITION

Proposition 2: For a polar code with block length $N = 2^n$, it can be decomposed into an outer code consisting of 2^k ($0 \le k \le (n-1)$) independent subcodes with block length 2^{n-k}, and an inner code consisting of 2^{n-k} independent subcodes with block length 2^k.

Proof: For a polar code $c_1^N = u_1^N\mathbf{B}_N\mathbf{F}_2^{\otimes n}$ with block length $N = 2^n$, its generation matrix $\mathbf{G}_N = \mathbf{B}_N\mathbf{F}_2^{\otimes n}$ can be decomposed as follows

$$\begin{aligned}\mathbf{G}_N &= \mathbf{B}_N\left((\mathbf{B}_{N/M}^{-1}\mathbf{G}_{N/M})\otimes(\mathbf{B}_M^{-1}\mathbf{G}_M)\right)\\ &= \mathbf{B}_N(\mathbf{B}_{N/M}^{-1}\otimes\mathbf{B}_M^{-1})(\mathbf{G}_{N/M}\mathbf{G}_M)\\ &= \mathbf{B}_N(\mathbf{B}_{N/M}^{-1}\otimes\mathbf{B}_M^{-1})(\mathbf{G}_{N/M}\otimes\mathbf{E}_M)(\mathbf{E}_{N/M}\otimes\mathbf{G}_M),\end{aligned} \tag{20}$$

where the second and the third equalities come from the fact that $(\mathbf{AB})\otimes(\mathbf{CD}) = (\mathbf{A}\otimes\mathbf{C})(\mathbf{B}\otimes\mathbf{D})$, and $\mathbf{E}_M$ denotes the $M\times M$ identity matrix where $M = 2^k$. By defining the permutation matrix $\Phi_N \triangleq \mathbf{B}_N(\mathbf{B}_{N/M}^{-1}\otimes\mathbf{B}_M^{-1})$, (20) can be recast as

$$\mathbf{G}_N = (\mathbf{E}_M\otimes\mathbf{G}_{N/M})\Phi_N(\mathbf{E}_{N/M}\otimes\mathbf{G}_M), \tag{21}$$

where the term on the L.H.S. of Φ_N indicates the outer code, which consists of $M = 2^k$ independent subcodes with block length $N/M = 2^{n-k}$; and the term on the R.H.S of Φ_N indicates the inner code, which consists of $N/M = 2^{n-k}$ independent subcodes with block length $M = 2^k$. ■

Remarks: The decomposition dose not change the polar codeword, however, we can exploit the independency of decomposed inner subcodes to concurrently compute the $M = 2^k$ soft input for adaptive OSD. And each adaptive OSD faces a decoding problem that is $1/M$ of original one. In this regard, the overall complexity is reduced. This motivates us to use adaptive OSD to decode outer subcodes, and use SC decoder to generate the soft messages for adaptive OSD.

B. CONCATENATED ADAPTIVE OSD

With polar decomposition, a concatenated adaptive OSD scheme is developed to further reduce the implementation complexity. We illustrate this scheme in Fig. 5.

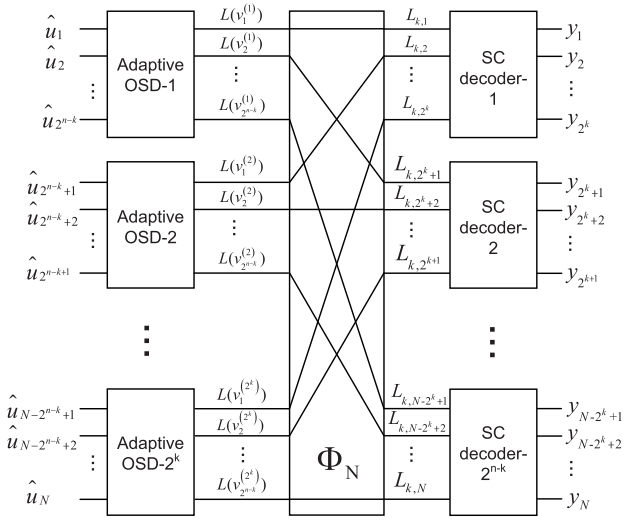


FIGURE 5. The concatenation of adaptive OSD.

For a received sequence y_1^N, its corresponding codeword is decomposed into 2^{n-k} inner subcodes and 2^k outer subcodes. SC decoders are used to generate message $L_{k,i}$ as the input to adaptive OSD ($L_{k,i}$ is the i-th LLR at level k in a data flow graph like Fig. 2). In particular, for a given SC decoder-j, ($1 \leq j \leq 2^{n-k}$), the LLR of its first bit can be computed given $y_{(j-1)\cdot 2^k+1}^{j\cdot 2^k}$. Due to the independency of inner subcodes, $\{L_{k,1}, L_{k,2^k+1}, \ldots, L_{k,N-2^k+1}\}$ can be obtained concurrently in one clock cycle. Under the permutation function Φ_N, we have $\{L(v_1^{(1)}), L(v_2^{(1)}), \ldots, L(v_{2^{n-k}}^{(1)})\} = \{L_{k,1}, L_{k,2^k+1}, \ldots, L_{k,N-2^k+1}\}$ as the soft input to the adaptive OSD-1 in Fig. 5. With Algorithm 1, the adaptive OSD-1 outputs $\hat{u}_1^{2^{n-k}}$, and the intermediate codeword $\{\hat{v}_1^{(1)}, \hat{v}_2^{(1)}, \ldots, \hat{v}_{2^{n-k}}^{(1)}\}$ can be computed. Using these intermediate codewords, the SC decoder-j ($1 \leq j \leq 2^{n-k}$) can compute the soft value of the second bit using $\hat{v}_j^{(1)}$ and $y_{(j-1)\cdot 2^k+1}^{j\cdot 2^k}$, which produces $\{L(v_1^{(2)}), L(v_2^{(2)}), \ldots, L(v_{2^{n-k}}^{(2)})\}$, and Adaptive OSD-2 could be activated in a similar manner as Adaptive OSD-1. Such process is continued until all information bits are decoded. This scheme is summarized as Algorithm 2.

We note that the first for-loop in line 2, it corresponds to the serial decoding process of 2^k concatenated adaptive OSDs, whereas the second for-loop corresponds to the decoding process for 2^{n-k} independent inner subcodes, which can be operated concurrently. Although the codeword dimension for some adaptive OSD-i is reduced from 2^n to 2^{n-k}, however, for each concatenated adaptive OSD component, it can only be activated until all its former component OSDs have derived their estimations. This increases decoding latency compared to Algorithm 1, thus there is a trade-off between decoding latency and complexity to implement the concatenated adaptive OSD.

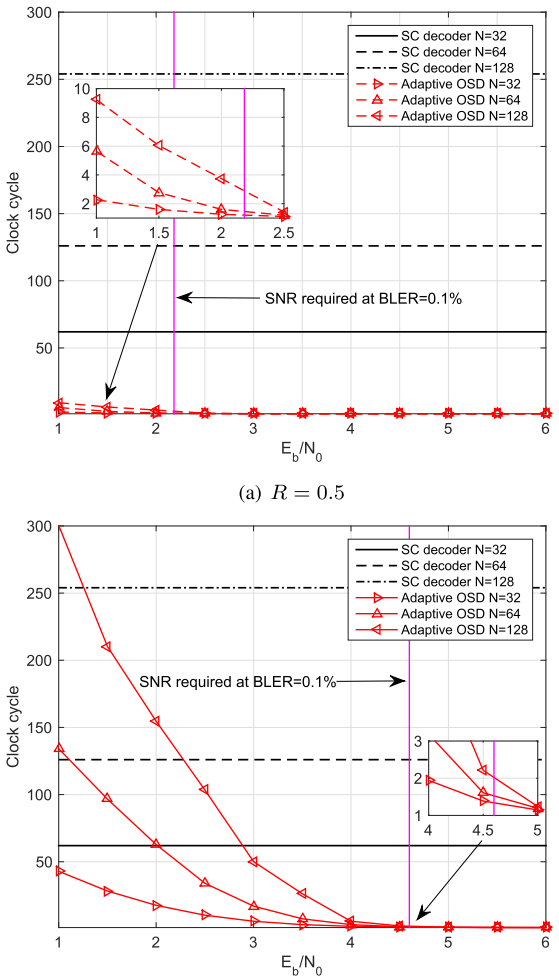
C. THE TRADE-OFF BETWEEN DECODING LATENCY AND COMPLEXITY

For all SC decoders in Fig. 5, they can be activated simultaneously. As introduced in Section II, the latency of SC decoding for a polar code with block length $M = 2^k$ is $(2M - 2)$ clock cycles, whereas the overall complexity of SC decoding part is $\frac{N}{M} \cdot O(M \log_2 M)$. For the adaptive

Algorithm 2 Concatenated Adaptive OSD Decoding

Input: k ($0 \leq k \leq (n-1)$), the received sequence y_1^N
Output: the estimated message $\hat{u}_1^N$

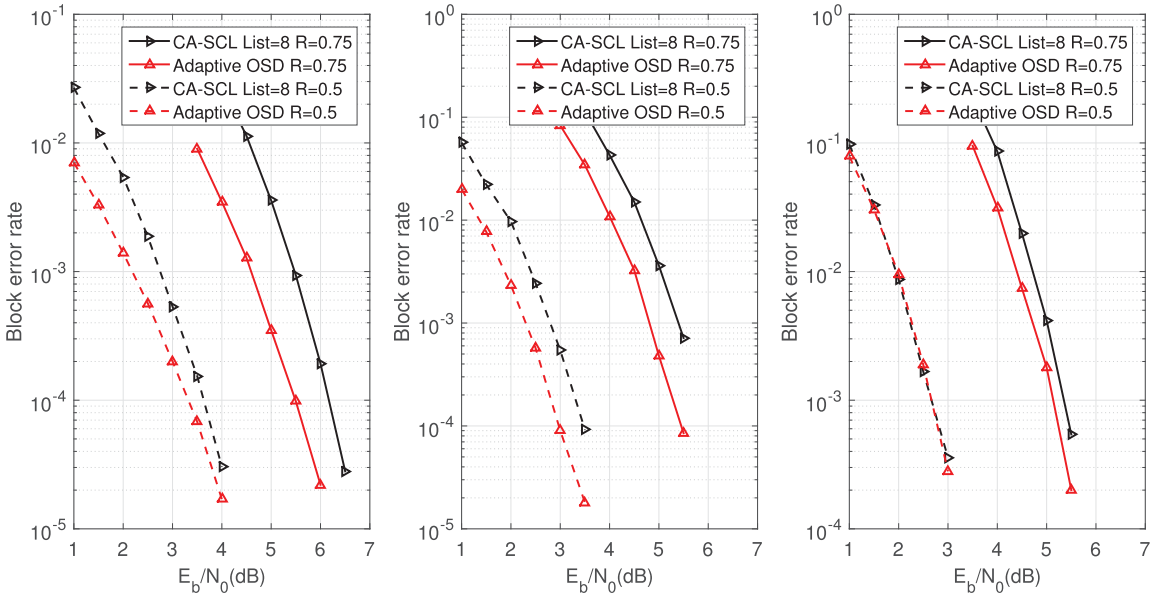
1 Decompose the current codeword into 2^{n-k} inner subcodes and 2^k outer subcodes.
2 **for** $i \leftarrow 1; i \leq 2^k; i \leftarrow i+1$ **do**
3 **for** $j \leftarrow 1; j \leq 2^{n-k}; j \leftarrow j+1$ **do**
4 $L(v_j^{(i)}) \leftarrow$ SC decoder-j($y_{(j-1)\cdot 2^k+1}^{j\cdot 2^k}, v_j^{(i-1)}$)
5 $u_{(i-1)2^{n-k}+1}^{i2^{n-k}} \leftarrow$ Adaptive OSD-i($L(v_1^{(i)}), L(v_2^{(i)}), \ldots, L(v_{2^{n-k}}^{(i)})$)
6 $\{v_1^{(i)}, v_2^{(i)}, \ldots, v_{2^{n-k}}^{(i)}\} \leftarrow u_{(i-1)2^{n-k}+1}^{i2^{n-k}} \cdot \mathbf{G}_{2^{n-k}}$
7 **return** $\hat{u}_1^N = \bigcup_{i=\{1,2,\ldots 2^k\}} u_{(i-1)2^{n-k}+1}^{i2^{n-k}}$



(a) $R = 0.5$

(b) $R = 0.75$

FIGURE 6. The Clock cycles required by adaptive OSD and SC decoder with block-length $N = 32, 64$ and 128 versus E_b/N_0 for coding rates $R = 0.5, 0.75$.



(a) N=32 (b) N=64 (c) N=128

FIGURE 7. The BLER achieved by the adaptive OSD and CA-SCL with a list size of 8 under coding rate $R = 0.5, 0.75$ versus E_b/N_0 for block-length $N = 32, 64$ and 128.

OSD part, its complexity is dominated by the sorting of received sequence and matrix diagonalization operations. Note that the size of input sequence for an adaptive OSD is reduced from N to N/M, and there are at most N/M information bits in $u_{(i-1)2^{n-k}+1}^{i2^{n-k}}$ $(i = 1, 2, \ldots 2^k)$ for each adaptive OSD, then we can estimate the decoding complexity for a level-0 adaptive OSD by $O(\frac{N}{M}\log_2\frac{N}{M}) + O(N/M)^3$. For a level-$l$ adaptive OSD, at most $\sum_{i=0}^{l}\binom{N/M}{i}$ clock cycles are required given sufficient hardware parallelization, and the proposed concatenated adaptive OSD works in a serial manner where the current OSD-i $(i = 1, 2, \ldots 2^k)$ must wait until OSD-1$\sim$OSD-(i-1) have decoded their intermediate codewords $\{v_1^{(1)}, \ldots, v_{2^{n-k}}^{(1)}, \ldots, v_1^{(i-1)}, \ldots, v_{2^{n-k}}^{(i-1)}\}$. Therefore, the decoding latency for all M adaptive OSD decoders can be estimated by $M \cdot \sum_{i=0}^{l}\binom{N/M}{i}$. By adding the latency of SC decoding part, the overall decoding latency of Algorithm 2 is at most $(2M-2) + M \cdot \sum_{i=0}^{l}\binom{N/M}{i}$.

Remarks: For each adaptive OSD component, the average clock cycle is strictly smaller than $\sum_{i=0}^{l}\binom{N/M}{i}$, and is close to 1 at high SNR regime (see Fig. 6). Thus the decoding latency of Algorithm 2 is positively correlated with the value of M. For lower decoding latency, smaller M value shall be selected. Specially, when $M = 2^0 = 1$, the Algorithm 2 degrades to the Algorithm 1. By choosing an appropriate M value, Algorithm 2 achieves a good trade-off between complexity and decoding latency.

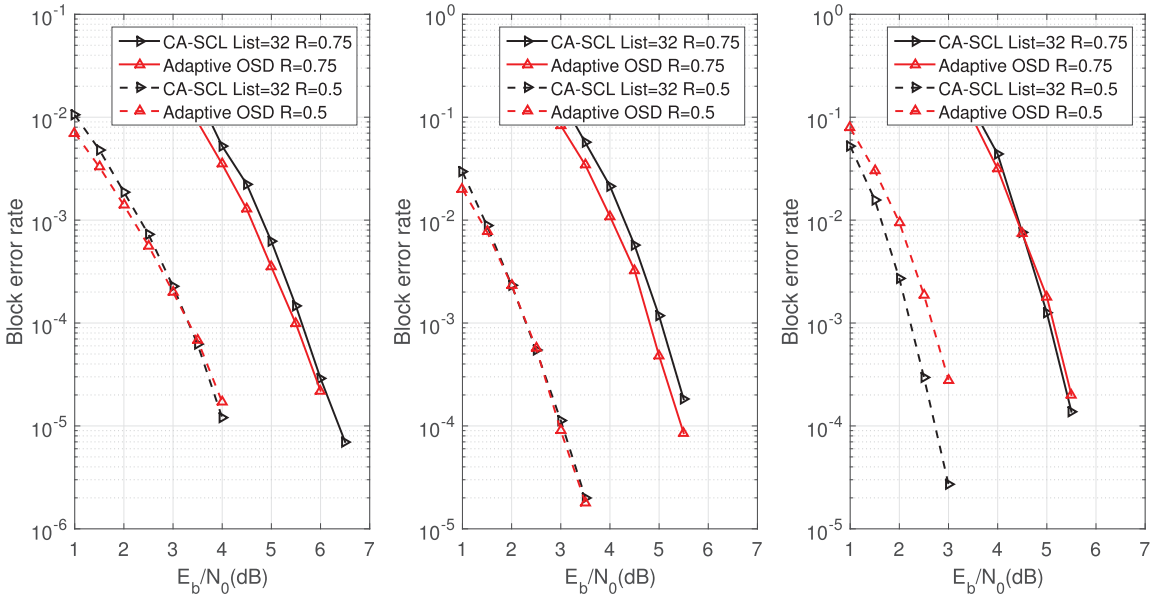
V. SIMULATION RESULTS AND COMPARISONS

In this section, we investigate two key performance indicators (KPIs) of polar decoding in URLLC, i.e., transmission reliability and decoding latency. The reliability feature is evaluated by the BLER performance of the proposed adaptive OSD, whereas the decoding latency is characterized by the consumed clock cycles. Like commonly deployed in URLLC transmissions, we use short block-length polar codes with medium to high coding rates. In particular, to construct polar codes, we adopt the off-line construction scheme introduced in [20]. Also, all polar codes are concatenated with a 8-bits CRC whose generator polynomial is $g(D) = D^8+D^5+D^4+1$ (see [21]). In this sense, the information rate of a codeword is $R' = \frac{K-8}{N}$ while the coding rate seen by a polar decoder is $R = K/N$.

A. DECODING LATENCY ANALYSIS

The average clock cycles used by the proposed adaptive OSD are numerically compared with conventional SC-based decoders under block-length $N = \{32, 64, 128\}$ and coding rates $R = \{0.5, 0.75\}$ in Fig. 6. To be more specific, we use standard SC decoder [3] as the benchmark for it has the consistent decoding latency compared to its other SC-based counterparts, e.g., CA-SCL decoders. In addition, the `max_level` in Algorithm 1 is set to 3 in this paper to meet the requirement of ultra-reliable transmission in URLLC.

For standard SC decoders, their decoding latency relate only to the block-length. However, the latency of Algorithm 1 relates not only to the block-length, but also to the working SNR and coding rate. For a certain coding rate, we can observe that the proposed adaptive OSD adapts to the current SNR in the sense that, more error patterns are tested when the working SNR is low, whereas for moderate or high SNR, the adaptive OSD can find the actual error pattern within much smaller clock cycles that are close to 1. On the other hand, for a certain working SNR, a higher coding rate means a larger search space, thus resulting more clock cycles.



(a) N=32 (b) N=64 (c) N=128

FIGURE 8. The BLER achieved by the adaptive OSD and CA-SCL with a list size of 32 under coding rate $R = 0.5, 0.75$ versus E_b/N_0 for block-length $N = 32, 64$ and 128.

Nevertheless, as the working SNR approaches to the decoding threshold where BLER $= 10^{-3}$ (as annotated in Fig. 6), the clock cycles used by the adaptive OSD drop dramatically, and even drop to that of "one-shot" decoding. This makes the proposed adaptive OSD practical in URLLC systems, where low SNR regime is actually not a regime of interest because the decoder will not be activated due to high BLER.

B. ERROR CORRECTING PERFORMANCE

We compare the BLER performance between the proposed decoder and the state of the art CA-SCL decoders in Fig. 7 and Fig. 8. Specially, list sizes of $L = \{8, 32\}$ are used in CA-SCL decoders to meet the requirement of ultra-reliable transmission.

In Fig. 7, the BLER performance achieved by the proposed adaptive OSD is compared with CA-SCL decoder with a list size of $L = 8$. One can observe that, the proposed adaptive OSD outperforms the CA-SCL decoder for various block-lengths under $L = 8$. Moreover, such gain becomes more obvious as the block-length getting smaller. It is reasonable since a smaller block-length means that a smaller number of erroneous hard decision will happen in the most reliable independent symbols. On the other hand, small block-length causes insufficient polarization of the split-channels for polar codes, which could increase its error probability under CA-SCL decoder. Additionally, the gap between CA-SCL decoder and our scheme under high coding rate is larger than that under moderate coding rates, which further indicates that the proposed adaptive OSD is more suitable for decoding polar codes with high coding rates and short block-lengths.

For larger list sizes, similar conclusions can be drawn from Fig. 8. However, the gap between CA-SCL and our scheme becomes narrow. This is practical, because for a CA-SCL decoder with a list size of 32, it yields the best BLER performance over its SC-based counterparts [8]. Nonetheless, for higher coding rates and shorter block-length cases, the proposed adaptive OSD still successfully competes the CA-SCL decoder, and therefore is more preferable for URLLC scenarios.

VI. CONCLUSION

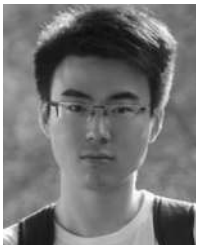
Capitalizing on the parallelizable nature of ordered statistic decoding, this paper firstly exploits a l-level OSD to avoid the data dependency involved in conventional SC-based decoding. Secondly, for the efficiency of hardware implementation, a novel codeword searching metric is developed, and we prove that the codeword with minimal metric value happens to be the codeword that minimizes the decoding error. For this reason, codewords with large metric value are considered to be unpromising and should be excluded. An adaptive OSD algorithm is then proposed following this sense. To further reduce the overall decoding complexity, we propose to divide the current codeword into several independent inner subcodes and outer subcodes. By handling these subcodes with concatenated adaptive OSD, a trade-off between decoding latency and complexity can be achieved. Finally, two key performance indicators of polar decoding in URLLC are evaluated through consumed clock cycles and BLER, respectively. We show that for messages with short block-length and high coding rate, the proposed adaptive OSD outperforms the state of the art CA-SCL decoders, and thus is more suitable for URLLC use case.

ACKNOWLEDGMENT

This article was presented in part at Globecom 2018 [1].

REFERENCES

[1] K. Qin and Z. Zhang, "Adaptive ordered statistic decoding of polar codes for URLLC systems," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–6.
[2] *Study on Scenarios and Requirements for Next Generation Access Technologies*, document TR 138 913, 3GPP, 2017.
[3] E. Arıkan, "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, Jul. 2009.
[4] *Chairmans Notes*, document 3GPP TSG RAN WG1 #87, 2016.
[5] Z. Zhang, K. Qin, L. Zhang, and G. T. Chen, "Progressive bit-flipping decoding of polar codes: A critical-set based tree search approach," *IEEE Access*, vol. 6, pp. 57738–57750, 2018.
[6] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aug. 2011, pp. 1–5.
[7] K. Chen, K. Niu, and J. R. Lin, "List successive cancellation decoding of polar codes," *Electron. Lett.*, vol. 48, no. 9, pp. 500–501, Apr. 2012.
[8] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.
[9] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.
[10] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency SC polar decoder architectures," in *Proc. IEEE ICC*, Jun. 2012, pp. 3471–3475.
[11] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2429–2441, May 2013.
[12] B. Li, H. Shen, D. Tse, and W. Tong, "Low-latency polar codes via hybrid decoding," in *Proc. 8th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Aug. 2014, pp. 223–227.
[13] X. Liu, J. Sha, C. Zhang, and Z. Wang, "A stage-reduced low-latency successive cancellation decoder for polar codes," in *Proc. IEEE Int. Conf. Digit. Signal Process. (DSP)*, Jul. 2015, pp. 258–262.
[14] Y. Fan, J. Chen, C. Xia, C.-Y. Tsui, J. Jin, H. Shen, and B. Li, "Low-latency list decoding of polar codes with double thresholding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1042–1046.
[15] Y. Fan, C. Xia, J. Chen, C.-Y. Tsui, J. Jin, H. Shen, and B. Li, "A low-latency list successive-cancellation decoding implementation for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 303–317, Feb. 2016.
[16] *Early Termination for Polar Codes*, document 3GPP TSGRAN WG1 Meeting #89, R1-1708644, Qualcomm, May 2017.
[17] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.
[18] D. Wu, Y. Li, X. Guo, and Y. Sun, "Ordered statistic decoding for short polar codes," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1064–1067, Jun. 2016.
[19] *Merge Sort*. Accessed: Jul. 2019. [Online]. Available: https://en.wikipedia.org/wiki/Merge_sort
[20] *Construction Schemes for Polar Codes*, document 3GPP TSG RAN WG1 Meeting #89, R1-1701702, Huaewi, HiSilicon, Feb. 2017.
[21] J. G. Proakis, *Digital Communications*. New York, NY, USA: McGraw-Hill, 1995.



KANGJIAN QIN received the B.S. degree in telecommunication engineering from the College of Telecommunication Engineering, Xidian University, Xi'an, China, in 2015. He is currently pursuing the Ph.D. degree with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. His research interests include 5G communications and channel coding.

ZHAOYANG ZHANG (M'02) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1998.

He is currently a Qiushi Distinguished Professor with Zhejiang University. He has coauthored more than 300 peer-reviewed international journal and conference articles, including seven conference best articles. His current research interests include the fundamental aspects of wireless communications and networking, such as information theory and coding, networked signal processing and distributed learning, AI-empowered communications and networking, and synergetic sensing, communication, and computation in the Internet of Things. He was awarded the National Natural Science Fund for Outstanding Young Scholars, in 2017. He is serving as an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON COMMUNICATIONS, and the *IET Communications* and has served as the General Chair, the TPC Co-Chair, or the Symposium Co-Chair for WCSP 2013/2018, the Globecom 2014 Wireless Communications Symposium, and the VTC-Spring 2017 Workshop HMWC. He also served as a Keynote Speaker for several international conferences and workshops, such as APCC 2018 and VTC-Fall 2017 Workshop NOMA.