IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xxxx

CRC-aided Logarithmic Stack Decoding of Polar Codes for Ultra Reliable Low Latency Communication in 3GPP New Radio

LUPING XIANG[1], ZEYNEP B. KAYKAC EGILMEZ[1], ROBERT G. MAUNDER[1], (Senior Member, IEEE) and LAJOS HANZO[1], (Fellow, IEEE)
[1]NGW Group, School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

L. Hanzo would like to acknowledge the financial support of the EPSRC projects EP/Noo4558/1, EP/PO34284/1, COALESCE, of the Royal Society's Global Challenges Research Fund Grant as well as of the European Research Council's Advanced Fellow Grant QuantCom.

ABSTRACT This paper provides a detailed tutorial on the Cyclic Redundancy Check (CRC)-aided Logarithmic Successive Cancellation Stack (Log-SCS) algorithm. We apply these algorithms for the ultra-reliable decoding of polar codes, which has relevance for the control channels of the Ultra-Reliable Low Latency Communication (URLLC) version of the Third Generation Partnership Project (3GPP) New Radio (NR). During the exploitation of the CRC codes to improve the error correction performance, we propose a novel technique which limits the number of CRC checks performed, in order to maintain a consistent error detection performance. Additionally, we propose a pair of techniques for further improving the performance of the Log-SCS polar decoder. We demonstrate that the proposed $S = 128$ Improved Log-SCS decoder achieves a similar error correction capability as a Logarithmic Successive Cancellation List (Log-SCL) decoder having a list size of $L = 128$ across the full range of block lengths supported by the 3GPP NR Physical Uplink Control Channel (PUCCH). This is achieved without increasing its memory requirement, while dramatically reducing its complexity, which becomes up to seven times lower than that of a $L = 8$ Log-SCL decoder.

INDEX TERMS NR polar codes, SCL, SCS, LLR

NOMENCLATURE

AWGN	Additive White Gaussian Noise
BLER	Block Error Rate
BP	Belief Propagation
CRC	Cyclic Redundancy Check
FAR	False Alarm Rate
LDPC	Low Density Parity Check
LLRs	Logarithmic-Likelihood Ratios
Log-SCL	Logarithmic Successive Cancellation List
Log-SCS	Logarithmic Successive Cancellation Stack
LRs	Likelihood Ratios
LTE	Long Term Evolution
NR	New Radio
PC	Parity Check
PDCCH	Physical Downlink Control Channel
PM	Path Metric
PPV	Polyanskyi-Poor-Verdù
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QPSK	Quaternary Phase Shift Keying
SC	Successive Cancellation
SCL	Successive Cancellation List
SCS	Successive Cancellation Stack
SNR	Signal to Noise Ratio
URLLC	Ultra-Reliable Low Latency Communication
XOR	eXclusive-OR
3GPP	Third Generation Partnership Project
5G	Fifth Generation

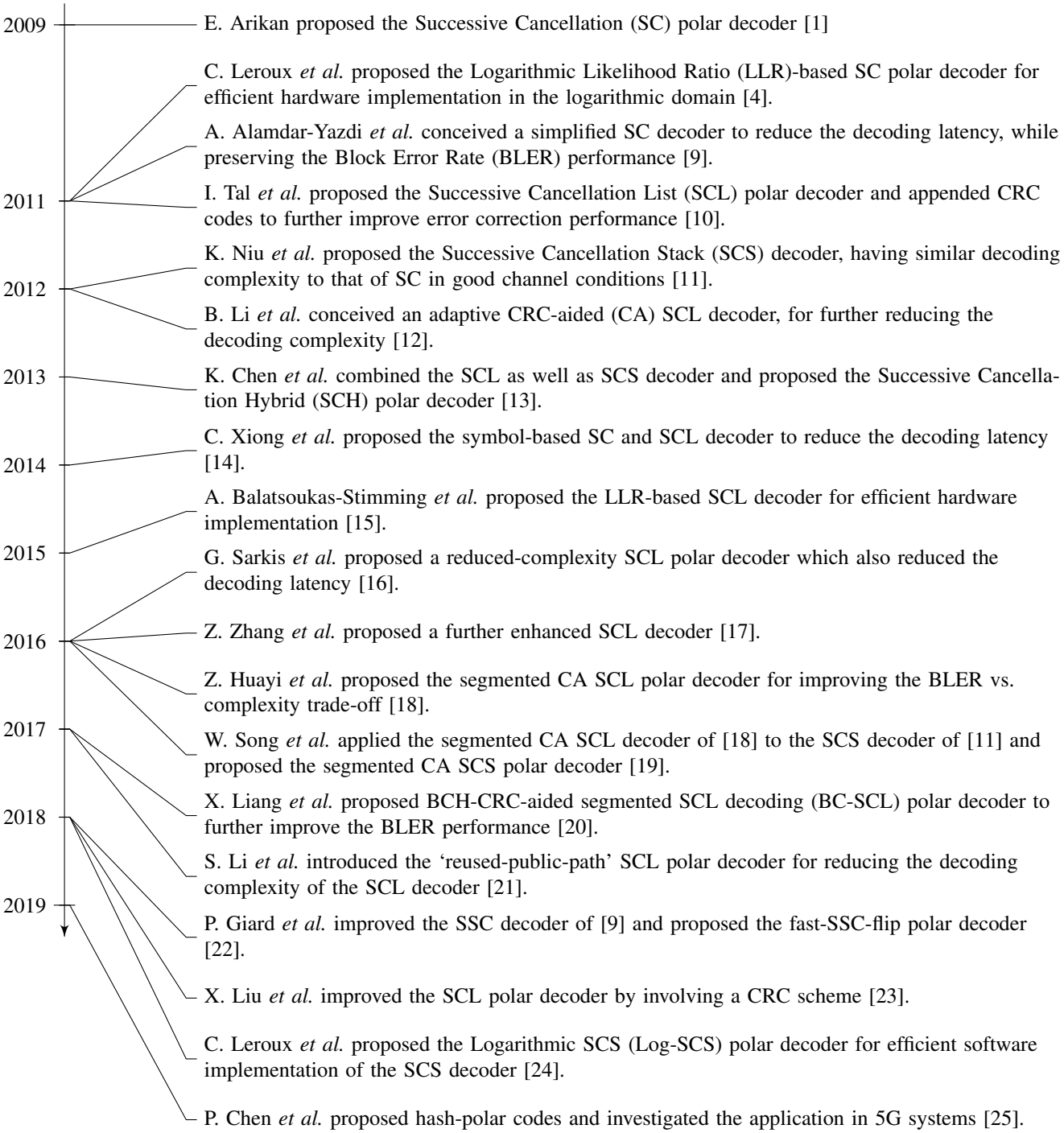


FIGURE 1: Timeline of the development of SC-based polar decoders.

I. INTRODUCTION

Owing to their near-capacity error correction performance at short block lengths, polar codes [1, 2] have been selected for protecting the control channels of the Third Generation Partnership Project (3GPP) New Radio (NR) candidate for fifth generation (5G) mobile communications [3]. The decoding algorithms of the polar codes fall into two categories, which either operate on the basis of Successive Cancellation (SC) [1, 4], or Belief Propagation (BP) [5–8]. Originally, the low-complexity SC [1, 4] decoding algorithm was proposed for polar codes, and was simplified in [9] to further reduce the complexity and the latency. However, more sophisticated decoding algorithms [10–25] are required for achieving near capacity performance in practical wireless channels, as summarised in Figure 1. Rather than considering only the locally most-likely value for each successive bit, as in SC decoding, the Successive Cancellation List (SCL) decoder [10, 26, 27] uses a breadth-first search for identifying the L number of locally most-likely bit values, allowing it to more frequently spot the globally most-likely values. By contrast, the Successive Cancellation Stack (SCS) decoder [11, 19] uses a depth-first approach to directly search for the globally most-likely values, although its ability to achieve this depends on the number of decoding candidates S that can be stored within the memory available for the stack. Note that while all L decoding candidates will grow to the same full length during the processing of the SCL algorithm at high channel Signal to Noise Ratios (SNRs), many of the S candidates in the SCS algorithm typically only reach much shorter lengths, hence reducing the complexity of the SCS algorithm below that of the SCL algorithm, approaching that of the SC algorithm. Typical benchmarkers of the SC-based decoders are listed in in Table 1.

The error correction performance of polar codes can be further improved by concatenating them with CRC codes [12, 20, 23, 27–34], as in 3GPP NR. During SCL or SCS decoding, the specific decoding candidates that do not satisfy the CRC can be discarded, even if this would otherwise appear to offer the globally most likely bit values. As an additional benefit, the CRC can enable error detection, although this ability is degraded by using the CRC to aid error correction in the manner described above. Furthermore, the distributed CRC technique allows early termination of the SCL or SCS decoding, further reducing the decoding complexity [18, 19, 35].

However, the conventional SCL [10] and SCS [11] decoders operate on the basis of bit probabilities, which have a high dynamic range and suffer from poor numerical stability, even in double precision floating point implementations. In order to address this problem, the authors of [15] proposed a Logarithmic SCL (Log-SCL) decoder that operates on the basis of Logarithmic-Likelihood Ratios (LLRs), which enables low-complexity fixed point implementation and reduced storage requirements, owing to the low dynamic range of LLRs [16, 36–38]. In addition, [24] proposed a Logarithmic-SCS (Log-SCS) decoder, which uses LLRs to bring the same advantages to the SCS decoder.

Motivated by this, we apply the Log-SCS algorithm to the polar code of the Physical Uplink Control Channel (PUCCH) and Physical Uplink Shared Channel (PUSCH) of the recently standardised Ultra-Reliable Low Latency Communication (URLLC) version of 3GPP NR [39]. Similar to [24], we provide a tutorial on how the Log-SCS algorithm may be obtained by transforming the computations of the SCS algorithm into the logarithmic domain, such that it can operate on the basis of LLRs. Furthermore, we provide a tutorial on how the Log-SCS algorithm improves on the SCS by considering the frozen bits, when determining the most likely sequence of information bits, which improves the error correction performance and reduces the decoding complexity. In addition, during the exploitation of the CRC codes to improve the error correction performance, we propose a novel technique for limiting the number of CRC checks performed in stack decoding, in order to maintain a consistent error detection performance. In this application, we demonstrate that the application of the Log-SCS decoder to the 3GPP NR polar code of PUCCH and PUSCH offers as much as 0.5 dB improved error correction performance, compared to the previous SCS algorithm using the same stack size of $S = 8$.

Additionally, we propose a pair of novel techniques for further improving the performance of the Log-SCS polar decoder. We show that across the full range of block lengths supported by NR PUCCH and PUSCH, the proposed $S = 128$ Improved Log-SCS decoder achieves a similar error correction capability as the $L = 128$ Log-SCL decoder. This is achieved despite dramatically reducing its complexity by up to seven times compared to a $L = 8$ Log-SCL decoder and without increasing its memory requirement. Owing to this, the proposed Improved Log-SCS decoder offers practical ultra-reliable error correction within as little as 0.5 dB of the channel capacity bound. Hence, it is particularly well-suited to the URLLC mode of 3GPP NR.

The rest of this paper is structured as follows. Section II provides a tutorial on the construction and the Log-SCS decoding of the 3GPP NR polar code for PUCCH and PUSCH. Following this, the performance vs. complexity and memory requirements of the CRC-aided Log-SCS decoder are compared to those of several benchmarkers in Section III. Then, our two novel improvements of the CRC-aided Log-SCS decoder are proposed in Section IV, where the error correction performance vs. complexity and memory resource requirement of the Improved Log-SCS decoder are also compared to those of the SCS and log-SCL decoders. Finally, we offer our conclusions and suggest avenues for future work in Section V.

II. OVERVIEW OF THE 3GPP NR UPLINK POLAR CODES

This section provides an overview of the 3GPP NR uplink polar codes [39], with a particular focus on the CRC-aided code used in the PUCCH and PUSCH channels, when the information block length is in the range of $A \in [20, 1706]$

TABLE 1: Summary of the error correction capability, computational complexity and memory requirement of various polar decoding algorithms. Specific measurements are provided for the example of $A = 84, G = 272$, where the error correction capability is measured by the EsN0 gap to the finite block-length capacity bound at a BLER of 10^{-3}, while the computational complexity is measured by the number of f, g and ϕ operations. In terms of the memory requirement, 1 byte is assumed for the storage of each bit probability and LLR.

	Novelty	Error correction capability	Computational complexity	Memory requirement
SC [1]	Originally proposed for polar decoding	poor (gap: 2.46 dB)	low (2048 operations)	low (0.56 KB)
SCL [10] (L=8)	Breadth-first search for L candidates	good (gap: 0.86 dB)	high (6.34 times higher than the SC)	medium (4.91 times higher than the SC)
SCS [11] (S=1024)	Depth-first search for S candidates	good (gap: 0.85 dB)	medium (3.17 times higher than the SC)	high (571.88 times higher than the SC)
Log-SCL [15] (L=128)	Extend the SCL to the logarithmic domain	good (gap: 0.70 dB)	high (174995 operations)	medium (40.25 KB)
Log-SCS [24] (S=1024)	Extend the SCS to the logarithmic domain	good (gap: 0.71 dB)	medium (21.31 times lower than the Log-SCL)	high (15.91 times higher than the Log-SCL)
Improved Log-SCS (S=128)	**Two techniques to further reduce the memory requirement and complexity**	good (gap: 0.71 dB)	low (41.67 times lower than Log-SCL)	medium (same as the Log-SCL)

bits[1]. Figure 2 illustrates the components of the 3GPP NR uplink polar encoder and decoder, each of which is discussed in the following subsections. More specifically, Sections II-A to II-F detail the code block segmentation and concatenation, CRC generation, appending and CRC check, frozen bit insertion and removal, polar encoder and decoder core operation, rate matching and dematching, as well as channel interleaving and deinterleaving, respectively.

A. CODE BLOCK SEGMENTATION AND CONCATENATION

If the information block length A and the encoded block length G satisfy $A \geq 1013$ OR ($A \geq 360$ AND $G \geq 1088$), then the information block input to the polar encoder of Figure 2 is first decomposed into two segments, as shown in Figure 3(a). Here, the first segment comprises the first $\lfloor A/2 \rfloor$ bits of the information block, but prepended with an additional zero-valued padding bit if A is odd, giving a total of $A' = \lceil A/2 \rceil$ bits. Meanwhile, the second segment is comprised of the remaining $A' = \lceil A/2 \rceil$ bits from the information block. The two information block segments are appended with CRC bits, inserted with frozen bits, encoded, rate matched and interleaved independently as discussed in the following sections. This results in a pair of encoded block segments, each comprising $E = \lfloor G/2 \rfloor$ bits, which are concatenated together and input to the modulator. If G adopts an odd value, then an additional zero-valued padding bit is appended to the block, to give a total of G bits. If $A \geq 1013$ OR ($A \geq 360$ AND $G \geq 1088$) is not satisfied, then the information block is processed as a single segment, as shown in Figure 3(a), with $A' = A$ and $E = G$. The corresponding inverse operations are performed in the polar decoder of Figure 2, where the padding bits are discarded.

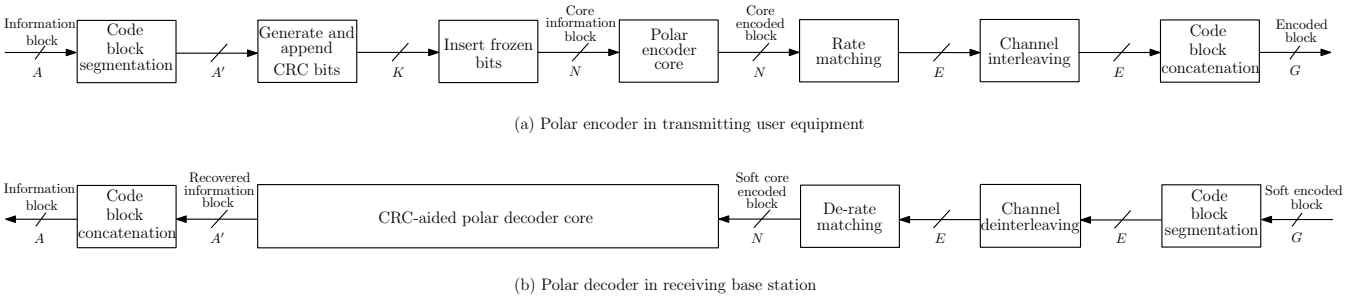
B. CRC GENERATION AND APPENDING

The polar code used in the NR uplink control channels for information blocks having lengths A in the range $[20, 1706]$ is protected by an 11-bit CRC code[2], having the generator polynomial $D^{11} + D^{10} + D^9 + D^5 + 1$. A set of 11 CRC bits are generated as a function of the A' information bits in each information block segment and are appended to give a sequence of $K = A' + 11$ bits [39], as shown in Figure 2.

Whenever the decoding core that will be detailed in Section II-D obtains a decoding candidate comprising a full set of N bits, then a CRC check may be performed in order to detect errors within the decoding candidate. If the CRC check succeeds, then the decoding candidate is output and the algorithm is terminated. However, if the CRC check fails, then a counter that has an initial value of zero is incremented and the decoding candidate is eliminated from the candidate sets. In this paper, we propose a novel technique of terminating the decoding algorithm if the counter reaches a value of eight, upon which a decoding

[1]Note that a Parity Check (PC) code is concatenated with the polar code in the case of PUCCH and PUSCH with information block lengths in the range $A \in [12, 19]$ bits [39]. We will apply our proposed Log-SCS to this case in future work.

[2]Note that a 6-bit CRC having the polynomial $D^6 + D^5 + 1$ is used to aid the PC polar code used for $A \in [12, 19]$ in 3GPP NR PUCCH and PUSCH.



(a) Polar encoder in transmitting user equipment

(b) Polar decoder in receiving base station

FIGURE 2: Schematic of the 3GPP NR uplink polar encoder and decoder.

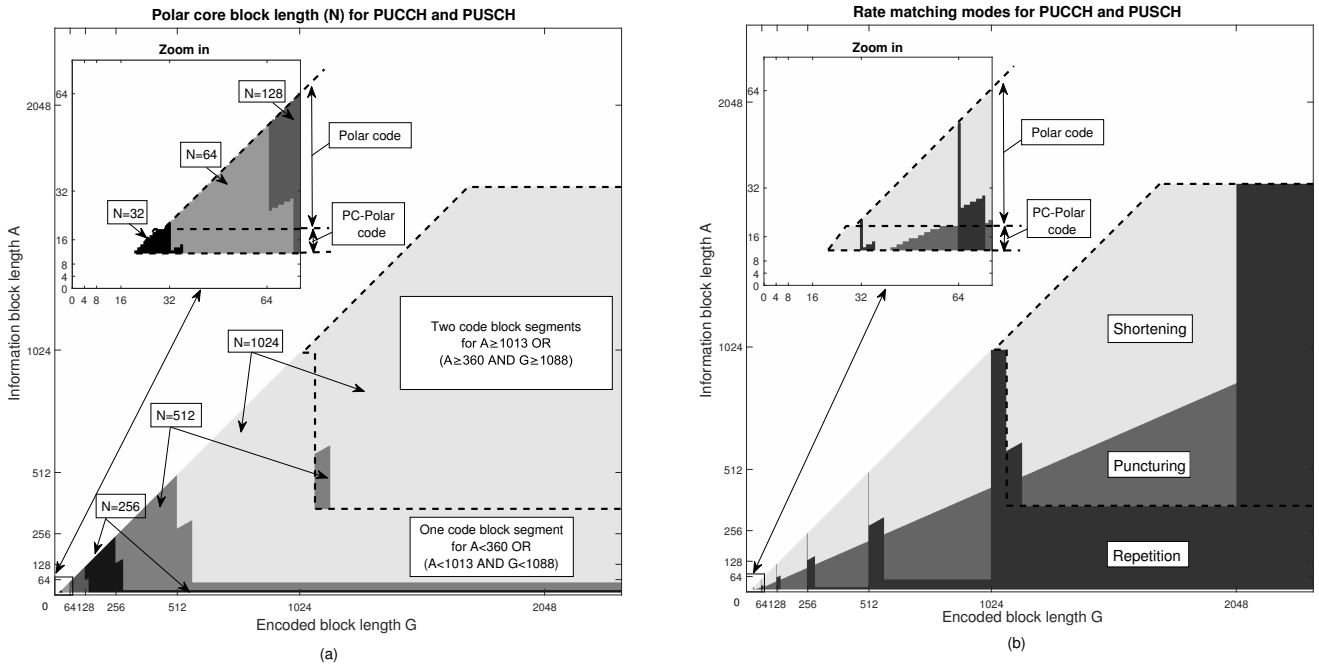


FIGURE 3: (a) Polar code block segmentation and core block length N for different combinations of information and encoded block lengths in PUCCH and PUSCH of 3GPP NR; (b) Rate matching modes for different combinations of information and encoded block lengths in PUCCH and PUSCH of 3GPP NR.

failure is declared. Note that this is in contrast to the CRC-aided SCS algorithm of [28], which continues considering successive CRCs indefinitely, until a pass is found, hence resulting in a high prevalence of undetected block errors.

C. FROZEN BIT INSERTION AND REMOVAL

Before invoking the polar encoder core, zero-valued frozen bits are inserted into the sequence of K information and CRC bits, in order to obtain a longer bit vector $\mathbf{u}$ comprising N bits, as shown in Figure 2. Here, N is a power of 2 that is higher than (or in principle, equal to) K, which may adopt values up to $N_{\max} = 1024$ in the NR PUCCH and PUSCH, as shown in Figure 3(b). Each of the N bits can be thought of as a polarised channel. Depending on the operation of the rate matching discussed in Section II-E and detailed in [39], the K most reliable channels are used to transmit the set of K information and CRC bits, while the frozen bits are mapped to the remaining $(N - K)$ least reliable channels [39]. In the receiver, the knowledge that the frozen bits have values of zero is exploited for aiding polar decoding, as will be detailed in Section II-D. Following polar decoding, the information bits are extracted from among the frozen bits, as shown in Figure 2.

D. POLAR ENCODING AND DECODING CORE

As shown in Figure 2, the vector $\mathbf{u}$ of $N = 2^n$ bits may be polar encoded into a vector $\mathbf{x}$ of N encoded bits, according to the modulo-2 matrix multiplication

$$\mathbf{x} = \mathbf{u}\mathbf{F}_2^{\otimes n}. \tag{1}$$

Here, $\mathbf{F}_2^{\otimes n}$ is the generator matrix, where the superscript $\otimes n$ represents the n^{th} Kronecker power of the kernel matrix $\mathbf{F}_2$,

which is given by

$$\mathbf{F}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

This operation may be represented graphically by the polar code graph of Figure 4, where the core information block is input on the left-hand edge of the graph and the successive stages of XOR operations produce the core encoded block on its right-hand edge.

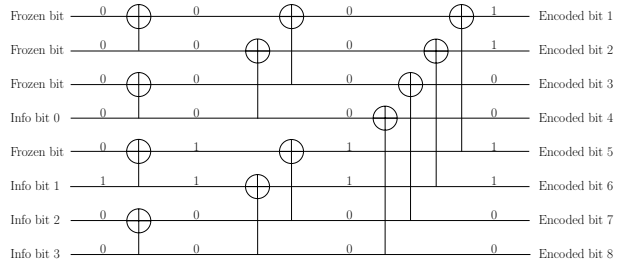


FIGURE 4: The XOR operations performed in the polar encoder core, for an example where $K = 4$ information bits [0100] are converted into $N = 8$ core encoded bits [11001100].

The corresponding decoding process of NR uplink polar codes may be completed by employing various algorithms including Log-SCL, SCS and Log-SCS. The operation of the Log-SCL decoding algorithm in the logarithmic domain may be applied to SCS decoding, by employing several modifications. The CRC-aided Log-SCS decoder is summarised in Algorithm 1 and compared to the SCS and Log-SCL algorithms in Figure 5. The Log-SCS decoder of [24] benefits from an improved error correction capability, despite its reduced complexity and other practical advantages compared to the CRC-aided SCS decoders of [11, 28]. These benefits are achieved in three ways. Firstly, the CRC-aided Log-SCS decoder of [24] operates on the basis of LLRs rather than the bit probabilities of the CRC-aided SCS decoders of [11, 28]. These LLRs have improved numerical stability and reduced dynamic range compared to bit probabilities, hence facilitating fixed point implementation and other practical benefits [16, 36, 37]. Furthermore, in contrast to the SCS algorithm, the Log-SCS algorithm considers not only information and CRC bits, but also frozen bits, when determining the most likely sequence of information bits, as shown in Figure 5(a). As we shall demonstarte in Section III, this decreases the decoding complexity and improves the error correction performance. As an additional refinement, this paper proposes the novel technique of limiting the number of CRC checks performed during the exploitation of the CRC codes to improve the error correction performance, in order to maintain a consistent error detection performance, as detailed below.

Like the Log-SCL decoder, the Log-SCS decoder operates on the basis of LLRs, where LLR $= \ln \frac{\Pr(\text{bit}=0)}{\Pr(\text{bit}=1)}$. These LLRs are combined in order to compute Path Metrics (PMs) that quantify the likelihood associated with a particular corresponding candidate sequence $[\hat{u}_i]_{i=1}^{N}$ of the N core information bits. Note that all frozen bits will adopt a value of 0 in any decoding candidate, but the information and CRC bits can adopt values of either 0 or 1. In both the Log-SCL and the Log-SCS decoders, the N bits in a decoding candidate are considered in sequential order, one at a time, with each successive bit updating the corresponding PM. During this process, the LLRs input at the right-hand edge of the polar code-graph of Figure 4 are combined by the XORs of the graph, as shown in Figure 6. More specifically, there are three types of computations that can be performed by a particular XOR in the graph, depending on the availability of LLRs provided on the connections at its right-hand side, as well as upon the availability of bits provided on the connections at its left-hand side [1].

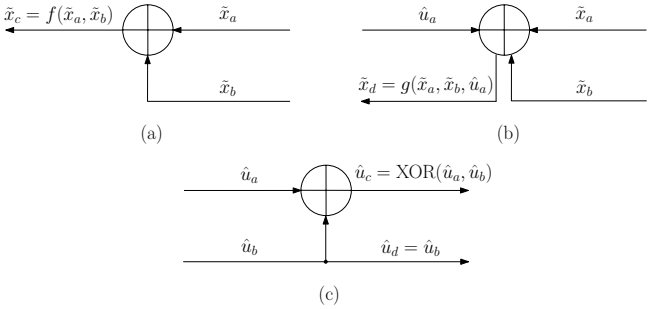


FIGURE 6: The three computations that can be performed for an XOR in the polar code graph: (a) the f function, (b) the g function and (c) partial sum calculation.

The first occasion when an XOR can contribute to the processing of a decoding candidate is when an LLR has been provided for both of the connections at its right-hand side, referred to as $\tilde{x}_a$ and $\tilde{x}_b$, respectively, as illustrated in Figure 6(a). This enables the XOR to compute a new LLR $\tilde{x}_c$ for the first of the two connections at its left-hand side, according to the f function, which can be expressed as

$$\begin{aligned} \tilde{x}_c &= f(\tilde{x}_a, \tilde{x}_b) \\ &= 2\tanh^{-1}\left(\tanh\left(\frac{\tilde{x}_a}{2}\right)\tanh\left(\frac{\tilde{x}_b}{2}\right)\right) && \text{(2a)} \\ &= \tilde{f}(\tilde{x}_a, \tilde{x}_b) + \log(1+\exp(-|\tilde{x}_a + \tilde{x}_b|)) \\ &\quad - \log(1+\exp(-|\tilde{x}_a - \tilde{x}_b|)) && \text{(2b)} \\ &\approx \tilde{f}(\tilde{x}_a, \tilde{x}_b), && \text{(2c)} \end{aligned}$$

where we have $\tilde{f}(\tilde{x}_a, \tilde{x}_b) \triangleq \text{sign}(\tilde{x}_a)\,\text{sign}(\tilde{x}_b)\min(|\tilde{x}_a|, |\tilde{x}_b|)$, and where (2b) is a numerically stable calculation of (2a), which may be further simplified to (2c).

Later in the decoding process, the hard bit decision $\hat{u}_a$ will be provided to the first of the two connections on the left-hand side of the XOR, as shown in Figure 6(b). Note that for the left-most XORs in the polar code graph, the corresponding hard bit decision will be provided by the corresponding bit of the decoding candidate under consideration. The hard bit decision $\hat{u}_a$ may be combined

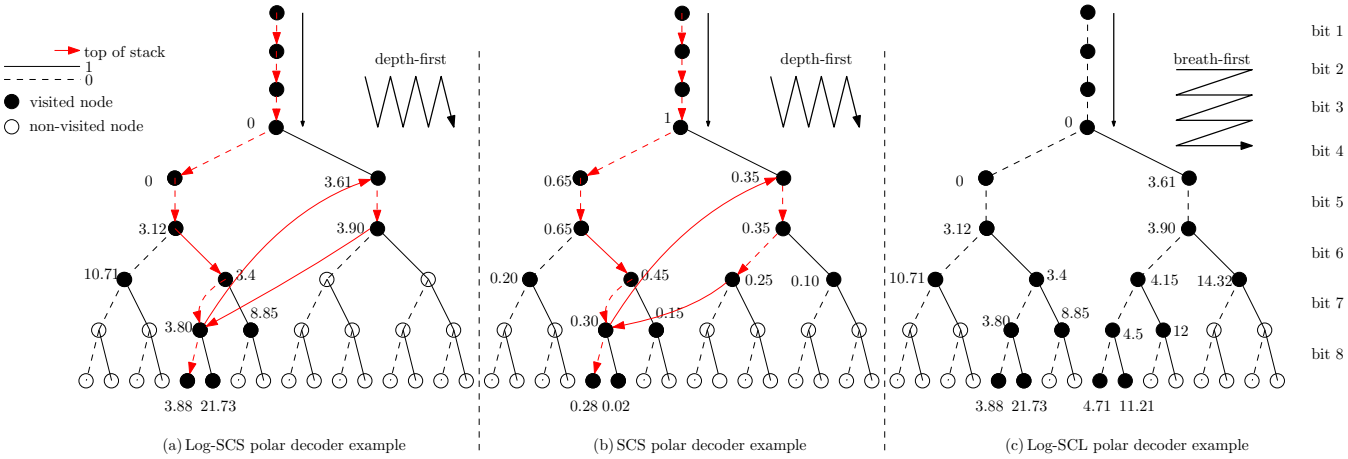


FIGURE 5: Code trees in Log-SCL, SCS and Log-SCS polar decoders, when decoding the example $K = 4$ information bits from the $N = 8$ core encoded bits of Figure 4, using the exact f and ϕ functions of (2a) and (5a).

with the LLRs $\tilde{x}_a$ and $\tilde{x}_b$ in order to compute a new LLR $\tilde{x}_d$ for the second connection on the left-hand side, according to the g function of

$$\begin{aligned}\tilde{x}_d &= g(\tilde{x}_a, \tilde{x}_b, \hat{u}_a) \\ &= (-1)^{\hat{u}_a}\tilde{x}_a + \tilde{x}_b. \end{aligned} \tag{3}$$

Later still, a bit $\hat{u}_b$ will be provided on the second of the connections on the left-hand side of the XOR, as shown in Figure 6(c). Again, this is provided by the corresponding bit of the decoding candidate, in the case of the left-most XORs. Together with $\hat{u}_a$, we can perform the partial sum computation of the bits $\hat{u}_c$ and $\hat{u}_d$ for the first and second connections on the right-hand side of the XOR, where we have

$$\begin{aligned}\hat{u}_c &= \text{XOR}(\hat{u}_a, \hat{u}_b) \\ \hat{u}_d &= \hat{u}_b. \end{aligned} \tag{4}$$

By performing the three types of XOR computations in a prescribed SC schedule [15], an LLR may be obtained for each of the N connections on the left-hand edge of the polar code graph, one at a time in sequential order from top to bottom. When the i^{th} LLR in this sequence $[\tilde{x}_i]_{i=1}^N$ is obtained, a PM may be updated for the decoding candidate according to [15]

$$\phi_i = \phi_{i-1} + \ln\left(1 + \exp\left[-\left(1 - 2\hat{u}_i\tilde{x}_i\right)\right]\right) \tag{5a}$$

$$\approx \begin{cases} \phi_{i-1}, & \text{if } \hat{u}_i = \frac{1}{2}\left[1 - \text{sign}(\tilde{x}_i)\right]; \\ \phi_{i-1} + |\tilde{x}_i|, & \text{otherwise,} \end{cases} \tag{5b}$$

where $\phi_0 = 0$. Here, the PM quantifies the likelihood of the decoding candidate, where having lower PMs implies higher likelihoods. Note that frozen bits contribute to the PM of both the Log-SCL and the Log-SCS algorithm, particularly when the corresponding LLR is negative. This is in contrast to the SCS algorithm of [11], where frozen bits do not impact the metrics used for selecting decoding candidates.

However, in contrast to the breadth-first approach of the Log-SCL algorithm, the Log-SCS algorithm adopts a depth-first approach. More specifically, the Log-SCL algorithm constructs a list of L candidate decoded bit sequences that are built one bit at a time in parallel, as shown in Figure 5(c). By contrast, the Log-SCS algorithm exemplified in Figure 5(a) considers only the most likely decoded bit sequence at a time, building it up one bit at a time, until it comprises N bits, or until its likelihood drops below that of another decoded bit sequence candidate, whereupon the focus switches to that candidate. More specifically, the Log-SCS decoder uses a stack to keep track of up to S decoding candidates at a time, where S is referred to as the stack size.

As shown in Algorithm 1, the Log-SCS begins with only a single decoding candidate in the stack, which initially has undefined values for all N bits. At each step of the Log-SCS algorithm, the decoding candidate at the top of the stack is selected and the value of its next undefined bit is considered. If this is an information or CRC bit, then the decoding candidate is updated to adopt a specific binary value for this bit and a replica is created that adopts the other binary value for this bit. The PMs for these two decoding candidates are updated and the replica is inserted into the stack. The decoding candidates are sorted in the order of their PM, with the top element in the stack having the lowest PM and the bottom element having the highest. If the insertion of the replica into the stack has resulted in exceeding the stack size S, then the bottom element in the stack is eliminated. By contrast, if the next undefined bit in the top decoding candidate is a frozen bit, then the bit in the decoding candidate is set to zero and no replica is made. The PM of the decoding candidate is updated and the stack is sorted in the order of increasing PM.

In contrast to the SCS algorithm shown in Figure 5(b), Figure 5(a) illustrates the advantage of updating the PMs with consideration of the frozen bits in the Log-SCS algorithm. In particular, when the decoding reaches the 7th bit, the top of the stack shifts from the decoding candidate

Algorithm 1 CRC-aided Log-SCS decoder

Input:
the received vector of N LLRs $\mathbf{y}$;
Output:
the decoded vector of N frozen, information and CRC bits $\hat{\mathbf{u}}$ when decoding is successful, or a NULL output when decoding is unsuccessful;
Initialization:
Initialise stack $\mathcal{S}$ with a single entry ℓ_{S_T};
Initialise corresponding PM $\phi[\ell_{S_T}] = 0$;
Initialise corresponding bit index $i[\ell_{S_T}] = 1$;
Initialise current stack size $s = 1$;
Initialise failed CRC counter $j = 0$;

while $j < 8$ and $s > 0$ **do**
  $i \leftarrow i[\ell_{S_T}]$;
  if $i \leq N$ **then**
    Use (2), (3) and (4) to compute the corresponding LLR $\tilde{x}_i[\ell_{S_T}]$ as a function of $\mathbf{y}$;
    if $i \notin \mathcal{A}$ **then**
      i is the index of a frozen bit;
      Set the corresponding bit $\hat{u}_i[\ell_{S_T}] = 0$;
      Use (5) to update the corresponding PM $\phi[\ell_{S_T}]$ as a function of $\tilde{x}_i[\ell_{S_T}]$ and $\hat{u}_i[\ell_{S_T}]$;
      Increment $i[\ell_{S_T}]$;
    else
      i is the index of a information or CRC bit;
      Create a replica of ℓ_{S_T}, referred to as ℓ_{S_B};
      Set the corresponding bits $\hat{u}_i[\ell_{S_T}] = 0$ and $\hat{u}_i[\ell_{S_B}] = 1$;
      Use (5) to update the corresponding PMs $\phi[\ell_{S_T}]$ and $\phi[\ell_{S_B}]$ as functions of $\tilde{x}_i[\ell_{S_T}]$, $\hat{u}_i[\ell_{S_T}]$ and $\hat{u}_i[\ell_{S_B}]$;
      Increment $i[\ell_{S_T}]$ and $i[\ell_{S_B}]$;
      Insert ℓ_{S_B} into the stack;
      Increment the stack size s;
      if $s > S$ **then**
        Remove the entry in the stack having the worst PM;
      end if
    end if
    Set ℓ_{S_T} to point to the entry in the stack having the best PM;
  else
    if CRC_Check($\hat{\mathbf{u}}[\ell_{S_T}]$) **then**
      return the decoded vector of N frozen, information and CRC bits $\hat{\mathbf{u}}[\ell_{S_T}]$;
    else
      Remove ℓ_{S_T} from the stack;
      Set ℓ_{S_T} to point to the entry in the stack having the best PM;
      Increment the failed CRC counter j;
    end if
  end if
end while
return NULL

with PM of 3.80 in Figure 5(a) or with likelihood of 0.30 in Figure 5(b), to the decoding candidate with PM 3.61 of the 4^{th} bit in Figure 5(a) or with likelihood of 0.35 in Figure 5(b). When processing the subsequent frozen 5^{th} bit, the Log-SCS algorithm in Figure 5(a) increases the PM to 3.90, but the SCS algorithm in Figure 5(b) simply copies the likelihood 0.35 of the previous bit. In Figure 5(a), the resultant PM of 3.90 is compared with the previous decoding candidate's PM of 3.80 and the Log-SCS returns to continue the decoding of its 8^{th} bit. By contrast, the SCS algorithm requires the additional consideration of the 6^{th} bit in the second decoding candidate, before it returns to complete the decoding of the first decoding candidate. In this way, the Log-SCS algorithm completes the decoding with a lower complexity than the SCS algorithm.

E. RATE MATCHING AND DEMATCHING

As shown in Figure 2, rate matching is applied after the polar encoder core, in order to adjust the length of the encoded block from N to E bits, where E does not have to be a power of 2. Here, $E \geq K$ must be satisfied, although E may be higher or lower than N. Here, rate matching for $E \geq N$ is achieved using repetition, while $E < N$ is achieved using puncturing or shortening. To be more specific, repetition repeats some of the N bits in the bit vector $\mathbf{x}$ [39], while shortening and puncturing remove some of the N bits in $\mathbf{x}$. Note that shortening removes some specific bits that are guaranteed to have values of 0, while puncturing removes some bits that may have either 0 or 1 values [39]. The selection between these three different rate matching modes in the NR uplink polar code depends on the combination of the information block length A and the encoded block length G [39], as shown in Figure 3(b).

In contrast to the bits used in the polar encoder, a logorithmic polar decoder operates on the basis of LLRs. In this case, the input to the rate dematching of the polar decoder shown in Figure 2 will be a vector of E LLRs. In the case of repetition, rate dematching is achieved by summing the LLRs corresponding to the repetition of each of the N bits in the vector $\mathbf{x}$, in order to obtain a corresponding vector of N LLRs. In the case of shortening, infinite-valued LLRs are inserted among the E LLRs in the positions of the shortened bits, in order to obtain the vector of N LLRs. By contrast, zero-valued LLRs are inserted in the positions of punctured bits, in order to represent the uncertainty over whether they have values of 0 or 1.

F. CHANNEL INTERLEAVING AND DEINTERLEAVING

Following the rate matching of Figure 2, the order of the E bits in the encoded block is rearranged by a channel interleaver, according to a prescribed triangular interleaving pattern of [39]. The corresponding deinterleaving operation is performed upon the E LLRs before they are subjected to the rate dematching in the polar decoder of Figure 2.

III. PERFORMANCE, COMPLEXITY AND MEMORY ANALYSIS

In the following subsections, the error correction and error detection performance, the computational complexity, and the memory requirement of the CRC-aided Log-SCS polar decoder of [24] for the 3GPP NR are characterised and compared to those of the SCS and Log-SCL algorithms of [11] and [15], in the context of PUCCH transmission. We investigate polar decoding employing QPSK modulation over an AWGN channel, since this is the channel model that was used throughout all of the 3GPP standardisation process and allows the direct comparison of our work to the large catalogue of results produced by 3GPP. Additionally, channel capacity bounds of the finite block length are available for AWGN channel allowing comparison with the theoretical limit. Some of our simulation results also consider uncorrelated Rayleigh fading channels, relying on the idealized simplifying assumption of perfect channel estimation, providing results for transmission over a more practical fading communication channel.

A. ERROR CORRECTION AND ERROR DETECTION PERFORMANCE

Figures 7 to 9 characterise the BLER performance of the polar decoders considered, when using QPSK modulation to convey blocks of various lengths over Additive White Gaussian Noise (AWGN) or Rayleigh channels. More specifically, an information block length of $A = 84$ bits is combined with encoded block lengths of $G = 136$, $G = 204$ and $G = 272$ bits, which respectively correspond to rate matching using shortening, puncturing and repetition, as shown in Figure 3(b). Here, the approximate f and ϕ computations of (2c) and (5b) are employed.

Observe in Figures 7 to 9 that superior BLER performance is achieved when higher encoded block lengths G are employed in both AWGN and flat Rayleigh fading channels, which is the result of two conflicting trends. Explicitly the coding-performance improves upon reducing the coding rate, which outweighs the effect of encountering more blocks having bit errors when extending the block length. As shown in the figures, the Log-SCS decoder achieves up to 0.5 dB gain compared to the SCS decoder having the same stack sizes. However, the Log-SCL polar decoder achieves the best BLER performance when the list size L of the Log-SCL decoder and the stack size S of the SCS and the Log-SCS decoders are both equal to 8. But as the stack size S is increased towards $S = 2048$, the BLER performance of the Log-SCS decoder becomes better than that of the $L = 8$ Log-SCL decoder, approaching the capacity bound within 1dB provided by the $\mathcal{O}(n^{-2})$ meta-converse Polyanskyi-Poor-Verdù (PPV) upper bound of [40], as shown in Figures 7 (a), 8 (a), and 9 (a). In contrast to the vertical Shannon capacity bound that is associated with the SNR where an infinitesimally low BLER may be achieved under the assumption of infinite block length, the PPV upper bound of [40] takes account of the finite block-length, in order to bound the achievable BLER as a function of SNR. The particular version of the PPV bound used here assumes QPSK modulation over an AWGN channel. Indeed, like the SCS decoder, the BLER performance of the Log-SCS decoder converges towards optimal maximum likelihood decoding, as the stack size S is increased towards infinity. However, an infinite stack size S leads to an infinite memory requirement, which is impractical for hardware implementations, as it will be detailed in Section III-C. Note that enhancements to the Log-SCS decoder will be proposed in Section IV, which address this performance discrepancy relative to the Log-SCL decoder.

Since the 3GPP PUCCH polar code employs a CRC having a limited length of $P = 11$ bits, there is a non-zero probability of the CRC failing to detect the erroneous bits in some blocks. The prevalence of this problem may be characterised by the False Alarm Rate (FAR), which can be defined as the fraction of blocks comprising random Gaussian distributed LLRs that nonetheless pass the CRC [41]. Our simulations have revealed that the FAR of the Log-SCS polar decoder remains near constant at around $2^{-(P-3)}$, which is consistent with that characterised in [41] for the Log-SCL decoder. Note that this FAR of $2^{-(P-3)}$ is maintained regardless of the stack size S, owing to our proposed approach of terminating the decoding when 8 failing CRCs have been encountered. Hence, this mechanism may be considered to use $\log_2(8) = 3$ CRC bits to aid error correction, while using the remaining $(P-3)$ CRC bits to perform error detection. Without this mechanism, the stack would continue offering new decoding candidates until eventually one with a passing CRC is found. However, the further down the stack this decoding candidate is found, the higher the probability of it containing erroneous bits. Hence, without the proposed approach for terminating the decoding, large stack sizes would lead to high FARs.

B. COMPUTATIONAL COMPLEXITY

In this section, we compare the computational complexity of the Log-SCS polar decoder to that of the Log-SCL decoder. Compared to the computation of the f, g and ϕ functions of (2), (3) and (5a), the partial sum calculation of (4) is of much lower computational complexity, which may be neglected in the complexity analysis of the algorithms considered. Therefore, Figures 10 and 11 show the total number of f, g and ϕ computations performed by the Log-SCS decoder, as well as by the Log-SCL decoder when employing an information block length of $A = 84$ and different encoded block lengths of $G = 136, 204$ or 272. As shown in Figures 10 and 11, the complexity of the Log-SCL decoder is independent of the channel SNR E_s/N_0, but the complexity of the Log-SCS decoder reduces in the SNR regions, where a low BLER is achieved. In these regions, the Log-SCS decoder has a complexity that is only one fifth of the $L = 8$ Log-SCL decoder's complexity regardless of stack size S. Note that the Log-SCS algorithm has a similar complexity to a decoder that uses the Log-SC algorithm in a first decoding

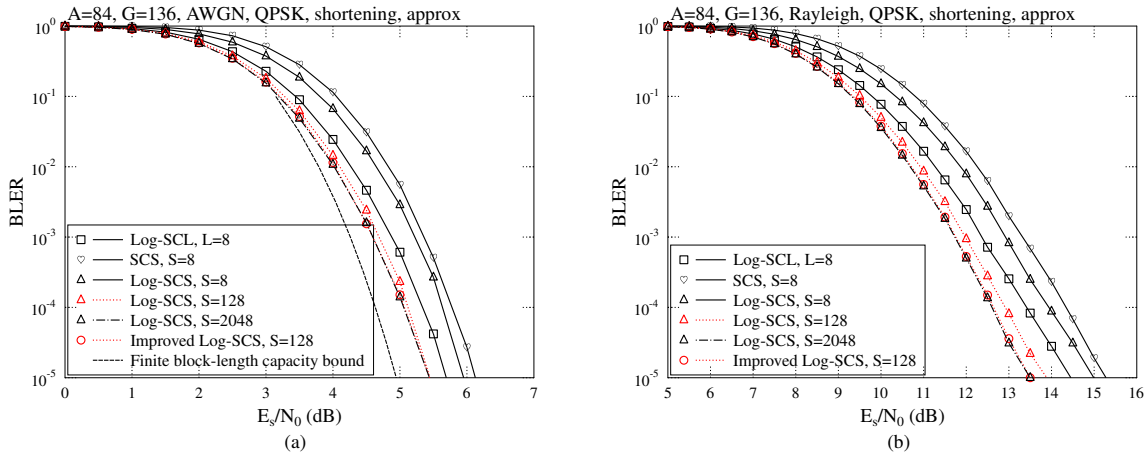


FIGURE 7: BLER performance of the polar decoders considered for $A = 84, G = 136$ polar codes with various stack sizes S, for the case where QPSK modulation is used for communication over (a) an AWGN channel; (b) a Rayleigh fading channel.

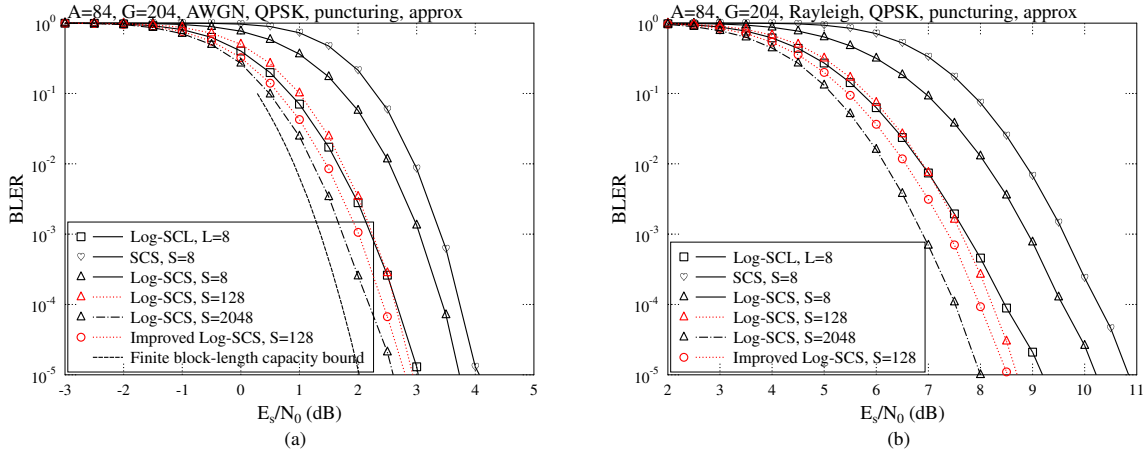


FIGURE 8: BLER performance of the polar decoders considered for $A = 84, G = 204$ polar codes with various stack sizes S, for the case where QPSK modulation is used for communication over (a) an AWGN channel; (b) a Rayleigh fading channel.

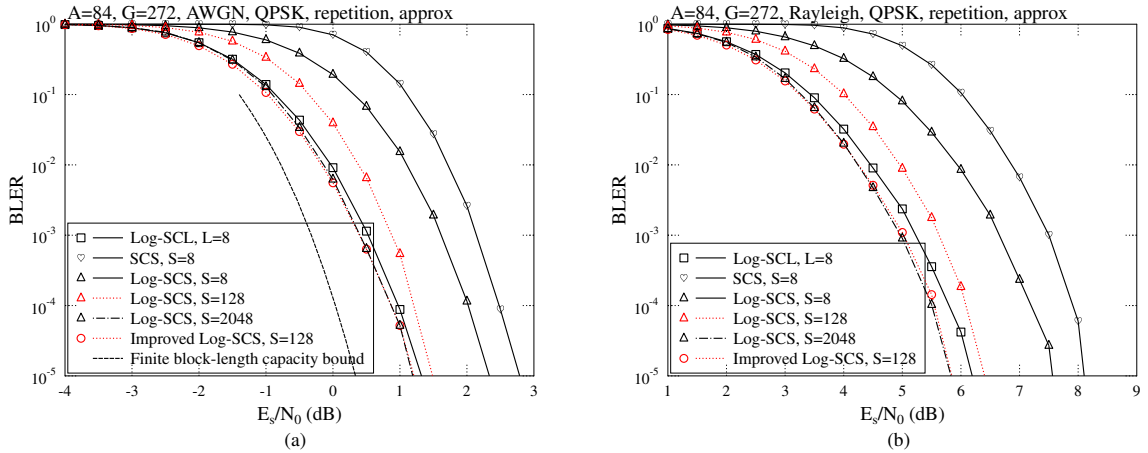


FIGURE 9: BLER performance of the polar decoders considered for $A = 84, G = 272$ polar codes with various stack sizes S, for the case where QPSK modulation is used for communication over (a) an AWGN channel; (b) a Rayleigh fading channel.

attempt and, only if that is unsuccessful, activates the Log-SCL algorithm having $L = 8$ in a second attempt. However, the Log-SCS algorithm has a lower processing latency, since it makes only a single decoding attempt and offers superior BLER when employing a sufficiently high stack size S, as shown in Section III-A.

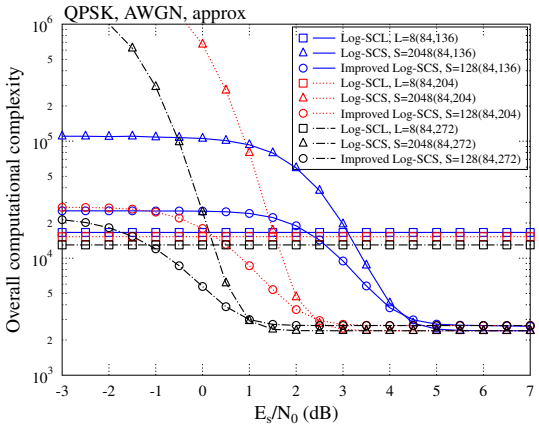


FIGURE 10: The number of f, g and ϕ functions performed by the polar decoders considered for various combinations of information block length A, encoded block length G and stack size S, for the case where QPSK modulation is used for communication over an AWGN channel.

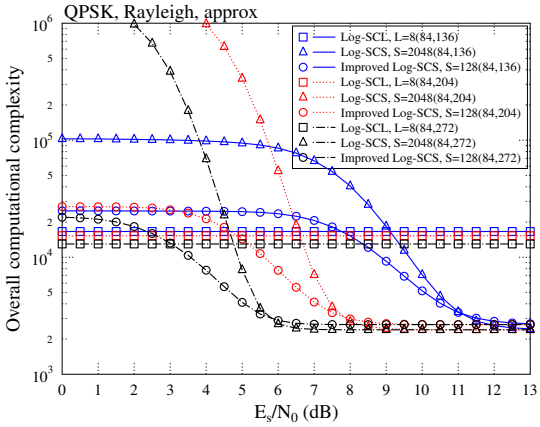


FIGURE 11: The number of f, g and ϕ functions performed by the polar decoders considered for various combinations of information block length A, encoded block length G and stack size S, for the case where QPSK modulation is used for communication over a flat Rayleigh fading channel.

C. MEMORY REQUIREMENT

In addition to the computational complexity, the hardware resource requirements of a practical polar decoder also depend on the maximum amount of memory required by the decoding algorithm, which comprises two parts. More specifically, memory is required for storing the partial sum bits processed by the g and XOR calculations of (3) and (4), as well as the LLRs processed by the f, g and ϕ calculations of (2), (3) and (5a).

For the Log-SCL decoder, memory is required for storing the $N_{\max}L$ candidate decoded bits that are generated during the decoding process, where the list size L quantifies the number of candidates and $N_{\max} = 1024$ is the maximum core block size used in 3GPP NR PUCCH and PUSCH. Furthermore, during the decoding process, the partial sum bits generated after each XOR operation must also be stored in memories. However, once we obtain the output bits of the XOR operations, the input bits are no longer needed. Therefore, memory is only required for $N_{\max}L$ partial sum bits during the Log-SCL decoding, comprising $N_{\max}$ bits for each of the L decoding candidates. So, the combination of the candidate decoded bits and the partial sum bits requires memory for a total of $2N_{\max}L$ bits is required by Log-SCL decoding, as shown in Table 2. By contrast, in Log-SCS decoding, the S decoding candidates may comprise different numbers of bits, as the decoding process proceeds. However, in the worst case, all S candidates will all comprise $N_{\max}$ bits. Hence, memory is required for $2N_{\max}S$ bits, including both the decoding candidate bits and the partial sum bits, as shown in Table 2.

The Log-SCL decoder requires $N_{\max}b_{\text{LLR}}$ bits to store the $N_{\max}$ LLRs input to the decoder, where b_{LLR} is the number of bits used for storing each LLR. The first decoding stage of the Log-SCL decoder requires storage for a further $LN_{\max}b_{\text{LLR}}/2$ bits, in order to represent the L LLR candidates output by each of the $N_{\max}/2$ f operations performed by this stage. Later on, this memory can be reused to store the L LLR candidates output by each of the $N_{\max}/2$ g operations performed by this stage. In this way, memory can be reused between the f and g operations performed by each stage in the polar code graph. Each successive stage requires half as much memory as the previous, requiring storage for a total of $(N_{\max} + LN_{\max}\sum_{i=1}^{\log_2 N_{\max}} 2^{-i})b_{\text{LLR}}$ bits for LLRs generated during the decoding process, as shown in Table 2. Similar logic can be applied when considering the worst case of the Log-SCS decoder, where at most $(N_{\max}+SN_{\max}\sum_{i=1}^{\log_2 N_{\max}} 2^{-i})b_{\text{LLR}}$ bits of storage is required for the LLRs, as shown in Table 2. Note that when $N_{\max}$ is sufficiently large, we have $\sum_{i=1}^{\log_2 N_{\max}} 2^{-i} \approx 1$.

Table 2 shows the example of the memory requirement for the considered polar decoders having $b_{\text{LLR}} = 8$ and $N_{\max} = 1024$, as in 3GPP NR PUCCH and PUSCH. Here, we can see that the $S = 128$ Log-SCS decoder and the $L = 128$ Log-SCL decoder require more than ten times as much memory as the $L = 8$ Log-SCL decoder.

IV. IMPROVEMENTS OF THE CRC-AIDED LOG-SCS POLAR DECODER

While the Log-SCS polar decoder achieves superior BLER vs. complexity performance compared to the $L = 8$ Log-SCL decoder, its requirement for a stack size of up to

TABLE 2: Memory requirements for the polar decoders considered

	Log-SCL		Log-SCS	Improved Log-SCS
LLR memory X (bits)	$(N_{\max} + LN_{\max}\sum_{i=1}^{\log_2 N_{\max}} 2^{-i})b_{\text{LLR}}$		$(N_{\max} + SN_{\max}\sum_{i=1}^{\log_2 N_{\max}} 2^{-i})b_{\text{LLR}}$	
Bit memory Y (bits)	$2N_{\max}L$		$2N_{\max}S$	
Overall memory $X+Y$ (bits)	$\approx (N_{\max} + LN_{\max})b_{\text{LLR}} + 2N_{\max}L$		$\approx (N_{\max} + SN_{\max})b_{\text{LLR}} + 2N_{\max}S$	
$N_{\max} = 1024$, $b_{\text{LLR}} = 8$, $X+Y$	$(L = 8)$11KB	$(L = 128)$161KB	$(S = 128)$161KB	

$S = 2048$ imposes a significantly higher memory requirement. Motivated by this, we propose two improvements to the Log-SCS decoder in Section IV-A and IV-B, allowing a smaller stack size S to be used, while still maintaining reliable low-complexity decoding. Following this, the error correction performance vs. complexity obtained when combining these two techniques is characterised and compared to the benchmarkers in Section IV-C.

A. REFERENCED LOG-SCS POLAR DECODER

Consider the example shown in Figure 12(a), where the first, second, third and fifth of the $N = 8$ bits are frozen. Here, the Log-SCS polar decoder fails to recover the correct bit sequence, which is associated with a PM of 3.88. As shown in Figure 12(a), this is because the path to the correct bit sequence is abandoned in the 10^{th} step of the algorithm, when its PM reaches 3.80, in favor of other paths having PMs of 3.70 and 3.75, but ultimately leading to incorrect bit sequences.

In order to address this situation, the Log-SCS algorithm may be further improved by preventing the longest path encountered so far from being removed from the stack. An example of this improvement is shown in Figure 12(b), where the Referenced Log-SCS decoder manages to decode and recover the original bit sequence by always retaining the longest path in the stack. In the example of Figure 12(b), the path having the metric 3.80 becomes the longest path encountered so far, and so it remains in the stack at the 10th step of the algorithm instead of being removed. This avoids the elimination of the correct path and achieves the correct decoding result. In this way, the Referenced Log-SCS decoder of Figure 12(a) can achieve a superior performance compared to the Log-SCS decoder, particularly in the case of a small stack size S.

B. RESTRICTED LOG-SCS POLAR DECODER

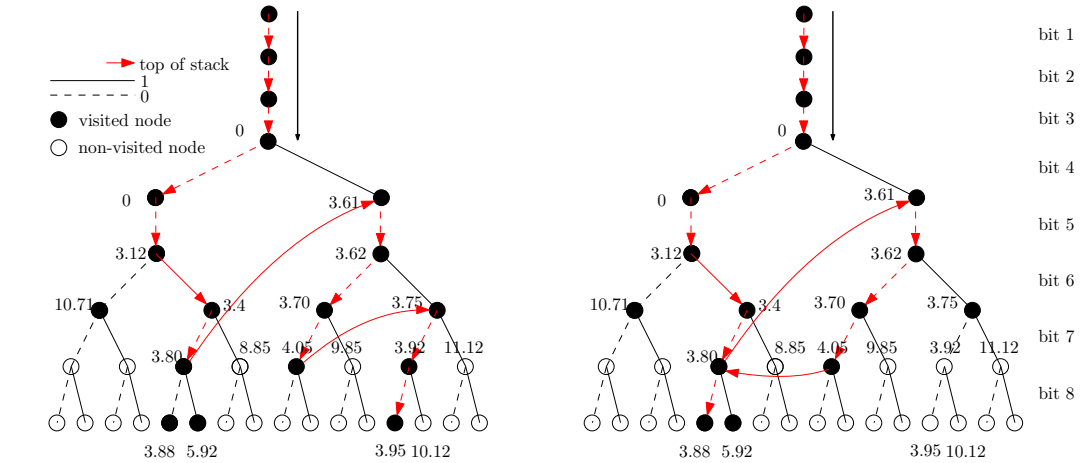
Another technique for maintaining the error correction performance despite having a reduced stack size S is to set a limit R for the maximum number of times that each of the N bits in the code tree may be visited during the process of traversing through the code tree. When this limitation is reached for any particular bit, the stack is pruned of all paths that have not yet reached this bit. In this way, more space can be released in the stack for storing paths that have longer path lengths, hence reducing the total memory required. Note that a larger R results in an improved error correction performance, but requires a higher stack size S, hence imposing a performance vs. complexity trade-off. In order to achieve a BLER similar to that of the $L = 128$ Log-SCL decoder while maintaining a reduced decoding complexity, we recommend $R = 32$ associated with $S = 128$.

C. PERFORMANCE OF THE IMPROVED CRC-AIDED LOG-SCS POLAR DECODER

The two refinements proposed in Section IV-A and IV-B may be combined for creating the Improved Log-SCS polar decoder. This section characterises the performance vs. complexity of the Improved Log-SCS polar decoder, as well as of the Log-SCL polar decoder, across the full range of block lengths supported by the 3GPP NR PUCCH and PUSCH polar code.

Figure 13 shows the SNR E_s/N_0 required by the polar decoders considered to obtain a BLER of 10^{-3}, across the full range of block lengths supported by the 3GPP NR PUCCH and PUSCH polar code, when using QPSK for communication over an AWGN channel. While Figure 13(a) employs the approximate f and ϕ functions of (2c) and (5b), Figure 13(b) is obtained using the exact expressions of the f and ϕ functions in (2b) and (5a). Here, the stack size of the Improved Log-SCS decoder is fixed to $S = 128$, whereas the list sizes of both $L = 8$ and 128 are considered for Log-SCL decoding. Additionally, the corresponding capacity bounds are provided by the $\mathcal{O}(n^{-2})$ meta-converse Polyanskyi-Poor-Verdù (PPV) upper bound of [40]. Compared to the approximate computations performed by the decoders of Figure 13(a), the exact decoders of Figure 13(b) require a lower E_s/N_0 for achieving a BLER of 10^{-3}, as may be expected. Both Figure 13(a) and (b) show that the $S = 128$-based Improved Log-SCS polar decoder consistently achieves a BLER of 10^{-3} at a similar E_s/N_0 as the $L = 128$ Log-SCL decoder. This is further illustrated in Figure 7 to 9, where the $S = 128$-based Improved Log-SCS polar decoder can be seen to offer a similar BLER to the $S = 1024$ Log-SCS decoder.

Figure 14 characterises the computational complexity gain (CCG) of the proposed $S = 128$-based Improved Log-SCS polar decoder across the full range of block lengths supported by the 3GPP NR PUCCH polar code. To be more specific, the CCG is defined as the ratio of the overall complexity of the $L = 8$ or $L = 128$-based Log-SCL decoder to that of the $S = 128$-based Improved Log-SCS decoder, at a BLER of 10^{-3}. Observe from Figure 14, that our proposed $S = 128$-based Improved Log-SCS decoder imposes a lower complexity than the $L = 8$-based Log-SCL



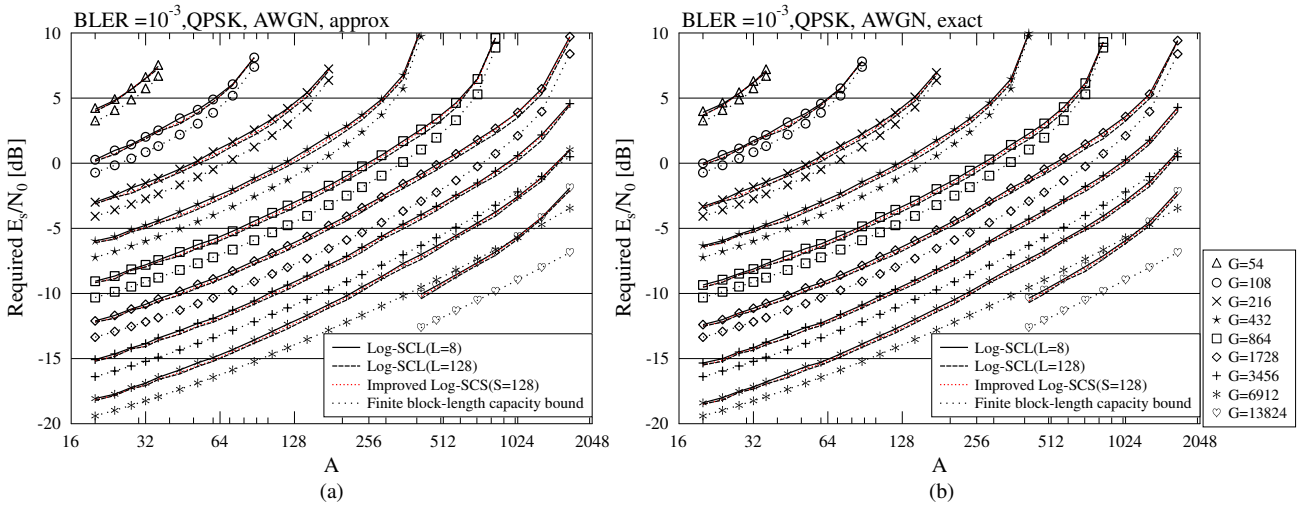
PMs inside the stack	Step t											
	1···3	4	5	6	7	8	9	10	11	12	13	14
0	—	0	3.12	3.4	3.80	3.61	3.62	3.70	4.05	3.75	3.92	3.95
		3.61	3.61	3.61	3.61	3.80	3.80	3.75	3.75	4.05	4.05	4.05

(a) Log-SCS

PMs inside the stack	Step t										
	1···3	4	5	6	7	8	9	10	11	12	13
0	—	0	3.12	3.4	3.80	3.61	3.62	3.70	4.05	3.80	3.88
		3.61	3.61	3.61	3.61	3.80	3.80	3.80	3.80		5.92

(b) Referenced Log-SCS

FIGURE 12: An example of code trees in (a) the original Log-SCS polar decoder and (b) Referenced Log-SCS polar decoder, where the stack size S is 2. Here, an $N = 8$-bit polar code is used, where the first, second, third and fifth of $N = 8$ bits are frozen. Here the exact of f and ϕ calculations of (2b) and (5a) are employed.



(a) (b)

FIGURE 13: The SNR required to achieve a BLER of 10^{-3} for different combinations of information block length A and encoded block length G of the 3GPP NR PUCCH polar code, using QPSK modulation for communication over an AWGN channel, when employing (a) the approximate f and ϕ functions of (2c) and (5b); and (b) the exact f and ϕ functions of (2b) and (5a). Here, the list size L for Log-SCL decoding is 8 or 128, whereas the stack size S for Log-SCS and Improved Log-SCS decoder is 128.

decoder across nearly all combinations of block length. The complexity of the proposed $S = 128$-based Improved Log-SCS decoder is up to 7 times lower than that of the $L = 8$ Log-SCL decoder and up to 100 times lower than that of the $L = 128$ Log-SCL decoder, while having the same memory requirement in the latter case.

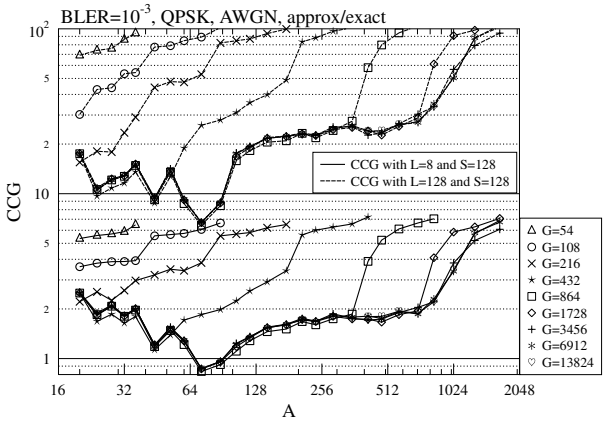


FIGURE 14: The computational complexity gain (CCG) for different combinations of information block length A and encoded block length G of the 3GPP NR PUCCH and PUSCH polar code to achieve a BLER of 10^{-3}, when employing QPSK for communication over an AWGN channel. Here, the list size L for Log-SCL decoding is 8 or 128, whereas the stack size S for the Improved Log-SCS decoder is 128.

V. CONCLUSIONS

In this paper, we have provided a detailed tutorial on the application of the Log-SCS algorithm to the 3GPP NR uplink polar code. We have demonstrated that the Log-SCS algorithm improves upon the BLER of the state-of-the-art Log-SCL polar decoder, while reducing the complexity. We have analysed the performance versus complexity and memory requirements in the context of PUCCH and PUSCH transmissions. During the exploitation of the CRC codes to improve the error correction performance, we have introduced the novel technique of limiting the number of CRC checks performed, in order to maintain a consistent error detection performance. Additionally, we have proposed two techniques for further improving the error correction performance of the Log-SCS polar decoder. We have compared the Improved Log-SCS decoder to the Log-SCL and SCS benchmarkers in terms of its BLER versus computational complexity and memory requirement. We have shown that the proposed $S = 128$-based Improved Log-SCS decoder achieves a similar error correction capability to the $L = 128$-based Log-SCL decoder, without increasing its memory requirement and while imposing only a complexity that is up to seven times lower than that of an $L = 8$-based Log-SCL decoder.

Our future work will focus on the application of the Improved Log-SCS decoder to the Physical Downlink Control Channel (PDCCH) transmissions in 3GPP NR, where a distributed CRC is used, as well as to the short block lengths of uplink PUCCH and PUSCH transmissions, where a PC code is concatenated with the polar code. Furthermore, our future work will consider the software and hardware implementation of the Improved Log-SCS polar decoder.

REFERENCES

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," IEEE Transactions on Information Theory, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] K. Niu, K. Chen, J. Lin, and Q. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," IEEE Communications magazine, vol. 52, no. 7, pp. 192–203, 2014.

[3] Huawei, HiSilicon, "R1-1608862: Polar code construction for NR," 3GPP TSG RAN WG1 Meeting #86bis, no. 10, 2016.

[4] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011, pp. 1665–1668.

[5] E. Arikan and E. Telatar, "On the rate of channel polarization," in 2009 IEEE International Symposium on Information Theory. IEEE, 2009, pp. 1493–1495.

[6] E. Arikan, "A performance comparison of polar codes and Reed-Muller codes," IEEE Communications Letters, vol. 12, no. 6, 2008.

[7] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," IEEE Transactions on Signal Processing, vol. 62, no. 24, pp. 6496–6506, 2014.

[8] Y. Zhang, A. Liu, X. Pan, Z. Ye, and C. Gong, "A modified belief propagation polar decoder," IEEE Communications Letters, vol. 18, no. 7, pp. 1091–1094, 2014.

[9] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," IEEE communications letters, vol. 15, no. 12, pp. 1378–1380, 2011.

[10] I. Tal and A. Vardy, "List decoding of polar codes," in Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on. IEEE, 2011, pp. 1–5.

[11] K. Niu and K. Chen, "Stack decoding of polar codes," Electronics letters, vol. 48, no. 12, pp. 695–697, 2012.

[12] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," IEEE Communications Letters, vol. 16, no. 12, pp. 2044–2047, 2012.

[13] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," IEEE Transactions on Communications, vol. 61, no. 8, pp. 3100–3107, 2013.

[14] C. Xiong, J. Lin, and Z. Yan, "Symbol-based successive cancellation list decoder for polar codes," in Signal Processing Systems (SiPS), 2014 IEEE Workshop on. IEEE, 2014, pp. 1–6.

[15] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," IEEE Transactions on Signal Processing, vol. 63, no. 19, pp. 5165–5179, 2015.

[16] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," IEEE Journal on Selected Areas in Communications, vol. 34, no. 2, pp. 318–328, 2016.

[17] Z. Zhang, L. Zhang, X. Wang, C. Zhong, and H. V. Poor, "A split-reduced successive cancellation list decoder for polar codes," IEEE Journal on Selected Areas in Communications, vol. 34, no. 2, pp. 292–302, 2016.

[18] H. Zhou, C. Zhang, W. Song, S. Xu, and X. You, "Segmented CRC-aided SC list polar decoding," in Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd. IEEE, 2016, pp. 1–5.

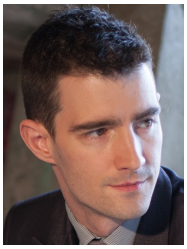
[19] W. Song, H. Zhou, Y. Zhao, S. Zhang, X. You, and C. Zhang, "Low-complexity segmented CRC-aided SC stack decoder for polar codes," in 2016 50th Asilomar Conference on Signals, Systems and Computers, Nov 2016, pp. 1189–1193.

[20] X. Liang, H. Zhou, Z. Wang, X. You, and C. Zhang, "Segmented successive cancellation list polar decoding with joint BCH-CRC codes," in 2017 51st Asilomar Conference on Signals, Systems, and Computers, Oct 2017, pp. 1509–1513.

[21] S. Li, L. Lu, Y. Deng, J. Liu, and T. Huang, "A Reused-Public-Path successive cancellation list decoding for polar codes with CRC," IEEE Communications Letters, vol. 21, no. 12, pp. 2566–2569, Dec 2017.
[22] P. Giard and A. Burg, "Fast-SSC-flip decoding of polar codes," in Wireless Communications and Networking Conference Workshops (WCNCW), 2018 IEEE. IEEE, 2018, pp. 73–77.
[23] X. Liu, S. Wu, X. Xu, J. Jiao, and Q. Zhang, "Improved polar SCL decoding by exploiting the error correction capability of CRC," IEEE Access, vol. 7, pp. 7032–7040, 2019.
[24] H. Aurora, C. Condo, and W. J. Gross, "Low-complexity software stack decoding of polar codes," in Circuits and Systems (ISCAS), 2018 IEEE International Symposium on. IEEE, 2018, pp. 1–5.
[25] P. Chen, B. Bai, Z. Ren, J. Wang, and S. Sun, "Hash-Polar codes with application to 5G," IEEE Access, pp. 1–1, 2019.
[26] K. Chen, K. Niu, and J. Lin, "List successive cancellation decoding of polar codes," Electronics letters, vol. 48, no. 9, pp. 500–501, 2012.
[27] I. Tal and A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, vol. 61, no. 5, pp. 2213–2226, 2015.
[28] K. Niu and K. Chen, "CRC-aided decoding of polar codes," IEEE Communications Letters, vol. 16, no. 10, pp. 1668–1671, 2012.
[29] U. U. Fayyaz and J. R. Barry, "Low-complexity soft-output decoding of polar codes," IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 958–966, 2014.
[30] T. Murata and H. Ochiai, "On design of CRC codes for polar codes with successive cancellation list decoding," in Information Theory (ISIT), 2017 IEEE International Symposium on. IEEE, 2017, pp. 1868–1872.
[31] Q. Zhang, A. Liu, X. Pan, and K. Pan, "CRC code design for list decoding of polar codes," IEEE Communications Letters, vol. 21, no. 6, pp. 1229–1232, June 2017.
[32] J. Chen, Y. Chen, K. Jayasinghe, D. Du, and J. Tan, "Distributing CRC bits to aid polar decoding," in 2017 IEEE Globecom Workshops (GC Wkshps), Dec 2017, pp. 1–6.
[33] F. Cheng, A. Liu, Y. Zhang, and J. Ren, "CRC location design for polar codes," IEEE Communications Letters, vol. 22, no. 11, pp. 2202–2205, Nov 2018.
[34] P. Chen, M. Xu, B. Bai, and J. Wang, "Design and performance of polar codes for 5G communication under high mobility scenarios," in Vehicular Technology Conference (VTC Spring), 2017 IEEE 85th. IEEE, 2017, pp. 1–5.
[35] D. Hui, M. Breschel, and Y. Blankenship, "Interleaved CRC for polar codes," in 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), June 2018, pp. 1–5.
[36] 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 Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015, pp. 1042–1046.
[37] J. Lin, C. Xiong, and Z. Yan, "A reduced latency list decoding algorithm for polar codes," in Signal Processing Systems (SiPS), 2014 IEEE Workshop on. IEEE, 2014, pp. 1–6.
[38] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 946–957, 2014.
[39] 3GPP TS 38.212 V15.1.1, "NR Multiplexing and channel coding," 3rd Generation Partnership Project Std. 3GPP, 2018.
[40] T. Erseghe, "Coding in the finite-blocklength regime: Bounds based on laplace integrals and their asymptotic approximations," IEEE Transactions on Information Theory, vol. 62, no. 12, pp. 6854–6883, 2016.
[41] Nokia, Shanghai-Bell, Alcatel-Lucent, "R1-1705860: Polar codes for control channels," 3GPP TSG-RAN WG1 #88bis Meeting, vol. 16, no. 10, 2017.

LUPING XIANG received the B.Eng. (Hons.) degree from Xiamen University, China, in 2015. He is currently pursuing the Ph.D. degree with the Southampton Wireless Group, University of Southampton. His research interests include channel coding and ultra low latency scheme.

ZEYNEP B. KAYKAC EGILMEZ received her dual BEng degrees in Electrical- Electronic Engineering and in Industrial Engineering from the Krkkale University, Republic of Turkey, in 2014. She graduated with first class honours, scoring the highest in the department of electrical and electronics engineering and the second highest in the engineering faculty, in 2014. She was awarded with a scholarship for MSc and Ph.D. degrees in UK, from the Turkish Ministry of National Education, in 2015. The University of Southampton has awarded her M.Sc. degree in wireless communications with distinction, Southampton, UK, in November 2017. Currently, she is working toward the Ph.D. degree in wireless communication with the University of Southampton. Her research interests include channel coding, especially polar coding in wireless communication.



ROBERT G. MAUNDER has studied with the School of Electronics and Computer Science, University of Southampton, UK, since October 2000. He was awarded a first class honours BEng in Electronic Engineering in July 2003, as well as a PhD in Telecommunications in December 2007. He began a lectureship in November 2007 and was promoted to Associate Professor in March 2013 and to Professor in August 2017. He was awarded Senior Member status of the IEEE in December 2012, Chartered Engineer status of the IET in November 2013 and Fellow status of the IET in January 2017. Rob's research interests include joint source/channel coding and the holistic design of algorithms and hardware implementations for wireless communications. He has published a number of IEEE papers in these areas. He is the founder and CTO of AccelerComm Ltd, which is commercialising his research as soft-IP.

LAJOS HANZO FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded an honorary doctorate by the Technical University of Budapest and in 2015 by the University of Edinburgh. In 2016 he was admitted to the Hungarian Academy of Science. During his 40-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised 111 PhD students, co-authored 18 John Wiley/IEEE Press books on mobile radio communications totalling in excess of 10 000 pages, published 1700+ research contributions at IEEE Xplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing a 60-strong academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European Research Council's Advanced Fellow Grant and the Royal Society's Wolfson Research Merit Award. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also a Governor of the IEEE VTS. During 2008 - 2012 he was the Editor-in-Chief of the IEEE Press and a Chaired Professor also at Tsinghua University, Beijing.