Low-Complexity Soft-Output Decoding of Polar Codes

Ubaid U. Fayyaz and John R. Barry

***Abstract*—The state-of-the-art soft-output decoder for polar codes is a message-passing algorithm based on belief propagation, which performs well at the cost of high processing and storage requirements. In this paper, we propose a low-complexity alternative for soft-output decoding of polar codes that offers better performance but with significantly reduced processing and storage requirements. In particular we show that the complexity of the proposed decoder is only 4% of the total complexity of the belief propagation decoder for a rate one-half polar code of dimension 4096 in the dicode channel, while achieving comparable error-rate performance. Furthermore, we show that the proposed decoder requires about 39% of the memory required by the belief propagation decoder for a block length of 32768.**

***Index Terms*—Polar codes, soft-output decoding, turbo equalization.**

I. Introduction

THE ERROR correcting code in a magnetic recording application must meet stringent error-floor and throughput requirements at a relatively large block length; the sector size for hard disk drives is typically 32768 bits, and the throughput can be 2 Gb/s or more. Regularity in the structure of the encoder/decoder facilitates hardware implementation by reducing interconnect congestion and processing requirements [1]. There has been significant research into finding regularly structured codes. For example, to avoid the high complexity of the early low-density parity check (LDPC) codes, which were random, different structured codes such as quasi-cyclic LDPC codes [2] emerged after their rediscovery in [3] and secured their place in different standards such as IEEE802.11n [4] and IEEE 802.16e [5].

An attractive alternative to LDPC codes are polar codes, discovered by Arikan [6], which feature a highly structured encoder and decoder that asymptotically achieve capacity on discrete memoryless channels. Additionally, they possess the desirable properties of universal rate adaptability, explicit construction and reconfigurability. As a result, they naturally demand further exploration for the long-block length throughput-limited magnetic recording channel. The first questions that arise are whether polar codes are a good fit for the magnetic recording application, and if they are, how well they perform. A typical detector architecture in magnetic recording channel relies on the turbo equalization principle [7] that iteratively exchanges soft information between a channel detector and an error-control decoder. To make polar codes feasible for such an iterative receiver, we need a decoder that can produce soft information about the coded bits. In this work, we propose a low-complexity soft-output decoder for polar codes that not only outperforms the existing soft-output decoder for polar codes, but also performs about 0.3 dB away from the belief propagation (BP) decoder with LDPC codes on a dicode channel for an FER $= 10^{-3}$.

In the seminal paper [6], Arikan proposed a hard-output successive cancellation (SC) decoder of complexity $O(N \log N)$, where N is the block length, that achieved capacity in the limit of large block lengths; its performance with finite-length codes was less promising. Since [6], improving the performance of the SC decoder has been at the forefront of the related research while the generation of soft information with reasonable complexity remained in background. In [8], the authors proposed a successive cancellation list decoder that performed better than the SC decoder at the cost of increased processing and storage complexity. They also showed that polar codes are themselves weak at small block lengths (e.g., $N = 2048$ or 4096), but if concatenated with a high-rate cyclic redundancy check (CRC) code (e.g., CRC-16), they perform comparably to state-of-the-art LDPC codes. Later in [9], the authors demonstrated that polar codes concatenated with CRC-24 codes can come within 0.2 dB of the information theoretic limit at as low a block length as $N = 2048$ using an adaptive successive cancellation list decoder with a very large list size. In [10], the authors proposed a successive cancellation stack decoder that also improved the performance over the SC decoder, but incurred huge storage requirements. Although all of these decoders offer better performance than the SC decoder, none provides the soft outputs essential for turbo-based receivers.

To the best of our knowledge, the only soft-output decoder for long polar codes that has appeared in the literature is a belief propagation (BP) decoder [11], [12]. The BP decoder has the advantages of having better performance than the SC decoder and providing soft outputs, but has very high storage and processing complexity. Consequently, the SC decoder remained an attractive choice for low-cost decoding of polar codes [11] for the applications that do not require soft outputs, and polar codes remained infeasible for turbo-based receivers. This work aims at making polar codes feasible for applications that require soft-output decoders. In particular, we develop a low-complexity soft-output version of the SC decoder called the *soft cancellation (SCAN)* decoder that produces reliability information for both the coded and message bits. SCAN

Manuscript received May 15, 2013; revised October 1, 2013 and December 10, 2013.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332–0250 (e-mail: ubaid@gatech.edu, john.barry@ece.gatech.edu).

Digital Object Identifier 10.1109/JSAC.2014.140515.

significantly reduces complexity. For example, the SCAN decoder requires only two iterations compared to 60 iterations of the BP decoder to achieve the same FER performance over a dicode channel and outperforms the BP decoder with further increase in the number of iterations. Furthermore, the SCAN decoder requires only $5N-2+\frac{N\log_2 N}{2}$ memory elements, significantly less than $2N(\log_2 N+1)$ memory elements required by the BP decoder.

The rest of the paper is organized as follows. In Section II, we describe the system model and the SC decoder. Section III explains the transition from the hard-output SC decoder to the soft-output SCAN decoder. The SCAN decoder in this form requires as many memory elements as the BP decoder does. Section IV demonstrates how can we reduce this huge storage requirement and propose the memory-efficient SCAN decoder. In Section VI, we present numerical results for the AWGN channel, the dicode channel and the EPR4 channel.

II. Preliminaries

A. System Model

We consider a polar code of length N, dimension K and construct it using the generator matrix $\mathbf{G}_N = \mathbf{G}_2^{\otimes n}$, where $n=\log(N)$, $(.)^{\otimes}$ denotes the nth Kronecker power and

$$\mathbf{G}_2 := \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{1}$$

We encode a message vector $\mathbf{m}=[m_0\ m_1,\ldots,\ m_{(K-1)}]$ of length K by first forming a vector $\mathbf{u}=[u_0\ u_1\ ,\ldots,\ u_{(N-1)}]$ such that $\mathbf{m}$ appears in $\mathbf{u}$ on the index set $\mathcal{I}\subseteq\{0,1,2,\ldots,N-1\}$ and then computing $\mathbf{v}=\mathbf{u\Pi G}_N$, where $\mathbf{\Pi}$ is a bit-reversal matrix as defined in [6]. In polar coding literature, the set $\mathcal{I}$ is usually referred to as the set of '*free indices*' and the complement $\mathcal{I}^c$ as the set of '*frozen indices*'. We set to zero the bits in $\mathbf{u}$ corresponding to the index set $\mathcal{I}^c$. The set $\mathcal{I}$ is known to both the encoder and the decoder. The construction of polar codes is equivalent to constructing $\mathcal{I}$; for a list of available construction methods, see [6], [13], [14] and [15]. We map $\mathbf{v}$ to $\mathbf{x}\in\{1,-1\}^N$ and pass the interleaved symbols $\mathbf{x}$ through a partial response channel impulse response $\mathbf{h}=[h_0h_1,\ldots,h_{\mu-1}]$ followed by an AWGN channel with noise variance $\sigma^2=N_0/2$, so that the k-th element of observation $\mathbf{r}$ at the output of the channel is

$$r_k=\sum_{i=0}^{\mu-1}h_i x_{k-i}+n_k, \tag{2}$$

where $n_k\sim\mathcal{N}(0,\sigma^2)$ is a Gaussian random variable with mean zero and variance σ^2. The per-bit signal-to-noise ratio is thus $E_b/N_0=\sum_i {h_i}^2/(2R\sigma^2)$, where $R=K/N$.

The receiver exchanges the soft information between the Bahl, Cocke, Jelinek and Raviv (BCJR) [16] channel equalizer and a soft-output polar decoder for some fixed number of iterations and iteratively estimates the transmitted message.

B. The SC Decoder

In [6], the authors proposed a successive cancellation (SC) decoder for polar codes. This decoder operates on a factor graph representation of polar codes that consists of $N(n+1)$ unique nodes, divided into $n+1$ columns indexed by $\lambda\in\{0,\ldots,n\}$. Each column consists of 2^λ groups indexed by $\phi\in\{0,\ldots,2^\lambda-1\}$, and each group consists of $2^{n-\lambda}$ nodes, represented by $\omega\in\{0,\ldots,2^{n-\lambda}-1\}$. Thus, we can pinpoint any node in this factor graph using the trio (λ,ϕ,ω). We define each of these groups, denoted by $\Phi_\lambda(\phi)$, as the set of nodes at a depth λ in the group ϕ. Additionally, the factor graph of polar codes contains a total of $2N-1$ such groups.

Fig.2 shows the factor graph of a rate-1/2 polar code of length $N=8$. This factor graph represents the relationship between encoded and uncoded bits. For the SC decoder, we construct two memory locations L and B of size $N(n+1)$, where $L_\lambda(\phi,\omega)$ is the log-likelihood value corresponding to the node, defined by the trio (λ,ϕ,ω). In this notation, $\{L_0(0,i)\}_{i=0}^{N-1}$ denotes the LLRs received from the channel. In the SC decoder, we estimate the message bits for all $\omega\in\{0,\ldots,N-1\}$ using

$$\hat{m}_i=\begin{cases}0 & \text{if } i\in\mathcal{I}^c \text{ or } L_n(i,0)\ge 0\\ 1 & \text{otherwise}\end{cases} \tag{3}$$

by going from $i=0$ to $N-1$ in increasing order, where $L_n(i,0)$ is computed using the recursion

$$L_\lambda(\phi,\omega)=L_{\lambda-1}(\psi,2\omega)\boxplus L_{\lambda-1}(\psi,2\omega+1) \tag{4}$$

for ϕ even, and

$$L_\lambda(\phi,\omega) = \begin{cases} L_{\lambda-1}(\psi,2\omega+1)+L_{\lambda-1}(\psi,2\omega) & \text{if } B_\lambda(\phi-1,\omega)=0,\\ L_{\lambda-1}(\psi,2\omega+1)-L_{\lambda-1}(\psi,2\omega) & \text{if } B_\lambda(\phi-1,\omega)=1,\end{cases} \tag{5}$$

when ϕ is odd, and $\boxplus$ is defined as

$$a\boxplus b\triangleq 2\tanh^{-1}\left[\tanh\left(\frac{a}{2}\right)\times\tanh\left(\frac{b}{2}\right)\right]. \tag{6}$$

Every time we calculate $L_n(i,0)$ for any $i\in\{0,1,\ldots,(N-1)\}$, we set $B_n(i,0)=\hat{m}_i$, where $\hat{m}_i$ is defined as in (3). When we calculate $L_\lambda(\phi,\omega)$ for odd values of ϕ, we update B using

$$B_{\lambda-1}(\psi,2\omega)=B_\lambda(\phi-1,\omega)\oplus B_\lambda(\phi,\omega), \tag{7}$$

$$B_{\lambda-1}(\psi,2\omega+1)=B_\lambda(\phi,\omega), \tag{8}$$

where $\oplus$ is binary XOR operation and $\psi=\lfloor\frac{\phi}{2}\rfloor$. For a detailed description of the SC decoder with pseudo-code, see [8].

III. The soft cancellation (SCAN) decoder

Consider the basic decision element of the factor graph in Fig.1 that represents polar codes of length two. Since in the SC decoder, all the processing on the factor graph of any polar code of length more than two occurs locally on this basic decision element, we can build our intuition and analysis on this factor graph of $N=2$ and then extend it to the general case of any N.

Suppose, we encode the bits u_0,u_1 using this polar code of length two, map them to $x_0,x_1\in\{+1,-1\}$ and send them on a binary-input DMC W with transition probabilities $W(y|x)$. The SC decoder first calculates the log-likelihood ratio for the bit u_0 using (4) with channel observations y_0,y_1 while assuming that u_1 is equally likely to be 0 or 1. The SC decoder assumes this about u_1, because it does not have an estimate of u_1 yet. Once it has an estimate for the bit u_0, it sets $B_1(0,0) = \hat{m_0}$ using (3) and calculates the log-likelihood ratio for the bit u_1 using (4) with the assumption that u_0 has been decoded with no error. After we calculate m_1 using $L_1(1,0)$ in (3), we set $B_1(1,0) = \hat{m_1}$ and use (5) to estimate the values of x_0, x_1. This final operation completes the SC decoding on this polar code of length two.

The aforementioned process transforms the vector channel $W_2(y_0, y_1|u_0, u_1)$ into two separate channels W_{SC}^- and W_{SC}^+ defined by the transition probabilities $W_{SC}^-(y_0, y_1|u_0)$ and $W_{SC}^+(y_0, y_1, u_0|u_1)$, respectively. We reiterate the assumptions used in the SC decoder as follows:

1) u_1 is equally likely to be 0 or 1 for the computation of likelihood $W_{SC}^-(y_0, y_1, u_1|u_0)$.
2) u_0 has been decoded with no error for the computation of likelihood $W_{SC}^+(y_0, y_1, u_0|u_1)$.

The first assumption is true only for very high E_b/N_0, whereas the second assumption is an oversimplification. Both of these assumptions distort LLR estimates, and we expect improved LLR estimates if we can incorporate soft information about u_0 and u_1 in the decoder instead of hard decision and no information, respectively. We first show in the following lemma how the likelihood computation changes if we have access to such soft information, and then we show how we provide this soft information in the SCAN decoder.

Fig. 1 explains the system model used in this lemma. We encode bits u_0 and u_1 to x_0 and x_1 and transmit on channel W. On the receiver, the SC decoder has y_0 and y_1 as channel observations to estimate transmitted bits. Now assume that we have information about u_0 and u_1 through other channels P_0 and P_1 in the form of z_0 and z_1, respectively. Lemma 1 describes the likelihood calculations for u_0 and u_1 given we have access to y_0, y_1, z_0 and z_1 if we follow the same order of detection as the SC decoder does.

Algorithm 1: The SCAN decoder

```
$\{L_0(0,i)\}_{i=0}^{(N-1)} \leftarrow$ LLRs from channel
$\{B_n(i,0)\}_{i\in\mathcal{I}^c} \leftarrow \infty,\ \{B_n(i,0)\}_{i\in\mathcal{I}} \leftarrow 0,$
$\{\mathcal{B}_\lambda(\phi)\}_{\lambda=0}^{n-1} \leftarrow 0,\ \forall\phi \in \{0,\ldots,2^\lambda - 1\}$
for $i = 1 \to I$ do
    for $\phi = 0 \to (N-1)$ do
        updatellrmap(n, $\phi$)
        if $\phi$ is odd then updatebitmap(n, $\phi$)
for $i = 0 \to (N-1)$ do
    if $(B_n(i,0) + L_n(i,0)) \geq 0$ then $\hat{m}_i \leftarrow 0$
    else $\hat{m}_i \leftarrow 1$
```

Lemma 1: Let $z_i : i \in \{0,1\}$ be the output of DMC's P_i, defined by the transition probabilities $P_i(z_i|u_i) : u_i \in \{0,1\}$ and conditionally independent of y_i. If we have access to z_i instead of perfect/no knowledge of u_i, the log-likelihood ratio of u_i under the SCAN decoder is given by

$$L_1(0,0) = L_0(0,0) \boxplus [B_1(1,0) + L_0(0,1)], \qquad (13)$$
$$L_1(1,0) = L_0(0,1) + [B_1(0,0) \boxplus L_0(0,0)]. \qquad (14)$$

The proof is provided in the Appendix.

Algorithm 2: updatebitmap(λ, ϕ)

```
if $\phi$ is odd then
    for $\omega = 0 \to (2^{n-\lambda} - 1)$ do
        $B_{\lambda-1}(\psi, 2\omega) \leftarrow B_\lambda(\phi-1,\omega) \boxplus [B_\lambda(\phi,\omega) + L_{\lambda-1}(\psi, 2\omega+1)]$   (9)
        $B_{\lambda-1}(\psi, 2\omega+1) \leftarrow B_\lambda(\phi,\omega) + [B_\lambda(\phi-1,\omega) \boxplus L_{\lambda-1}(\psi, 2\omega)]$   (10)
    if $\psi$ is odd then updatebitmap($\lambda - 1, \psi$)
```

Algorithm 3: updatellrmap(λ, ϕ)

```
if $\lambda = 0$ then return $\psi \leftarrow \lfloor \frac{\phi}{2} \rfloor$
if $\phi$ is even then updatellrmap($\lambda - 1$, $\psi$) for
$\omega = 0 \to (2^{n-\lambda} - 1)$ do
    if $\phi$ is even then
        $L_\lambda(\phi,\omega) \leftarrow L_{\lambda-1}(\psi, 2\omega) \boxplus [L_{\lambda-1}(\psi, 2\omega+1) + B_\lambda(\phi+1,\omega)]$   (11)
    else
        $L_\lambda(\phi,\omega) \leftarrow L_{\lambda-1}(\psi, 2\omega+1) + [L_{\lambda-1}(\psi, 2\omega) \boxplus B_\lambda(\phi-1,\omega)]$   (12)
```

The only problem remains now is to show how can we provide these additional LLRs $B_1(0,0), B_1(1,0)$ in all decision elements in a factor graph for any N. In the start of a decoding cycle, we compute $\{L_0(0,k)\}_{k=0}^{(N-1)}$ as we receive symbols $\mathbf{r}$ from the channel. We inform the decoder about the location of fixed bits by initializing $\{B_n(k,0)\}_{k\in\mathcal{I}^c}$ to ∞. Suppose, we are interested in finding the LLR $L_n(i,0)$ in (4) and (5) with $\{L_n(k,0)\}_{k=0}^{(i-1)}$ already computed and no information about $\{u_k\}_{k=(i+1)}^{N-1}$. Since we cannot have any information about $\{u_k\}_{k=(i+1)}^{N-1}$ in the first iteration, we will keep the assumption that they are equally likely, i.e., $B_n(k,0) = 0, \forall (i+1) \leq k \leq (N-1)$. It is noteworthy that we have already populated L partially from left to right while calculating $\{L_n(k,0)\}_{k=0}^{(i-1)}$. Therefore, as we calculate $\{L_n(k,0)\}_{k=0}^{(i-1)}$, we can use the partially calculated L as a-priori information to update B from right to left using (13) and (14) on all the decision elements involved. When $i = N-1$, we have B with extrinsic LLRs corresponding to all the nodes in the decoder's factor graph.

We once again start computing LLRs $\{L_n(i,0)\}_{i=1}^{N-1}$, but this time we have soft information in B for $\{u_k\}_{k=(i+1)}^{N-1}$ unlike the first iteration. Therefore, we can use B to supply a-priori information to all decision elements in subsequent iterations. We use this iterative process I times and use the extrinsic LLRs $\{L_n(i,0)\}_{i=0}^{N-1}$ and $\{B_0(0,i)\}_{i=0}^{N-1}$ calculated in the last iteration corresponding to message and coded bits, respectively. We explain all the necessary implementation details in Algorithms 1, 2 and 3.

Algorithm 1 provides the decoder's wrapper and calls

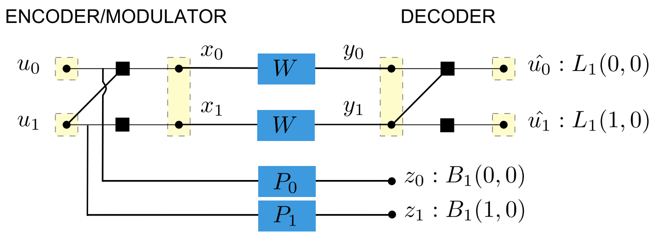


Fig. 1. System model for Lemma: 1

Algorithm 3 to calculate $\{L_n(\phi,0)\}_{\phi=0}^{(N-1)}$. Algorithm 3 updates L from left to right using B as prior information. Since B is initialized to zero except $\{\mathcal{B}_n(i,0)\}_{i=0}^{(N-1)}$, $B_\lambda(\phi+1,\omega)$ in (11) has zero value in the first iteration, just like the SC decoder. On the other hand, the SCAN decoder in the first iteration uses soft information in $B_\lambda(\phi-1,\omega)$ for (12) in contrast to the SC decoder which uses hard information about $B_\lambda(\phi-1,\omega)$ in (5). As we iterate through ϕ in the inner loop of Algorithm 1, for the odd values of ϕ the wrapper calls Algorithm 2 to update B from right to left. Algorithm 2 populates B using L as prior information, and by the end of the first iteration, $\{B_0(0,\phi)\}_{\phi=0}^{(N-1)}$ contains extrinsic LLRs for the coded bits. In the second iteration, (11) uses the values of $B_\lambda(\phi+1,\omega)$ from the first iteration, unlike the first iteration in which $B_\lambda(\phi+1,\omega)$ were initialized to zero. Algorithm 1 repeats this process for I times using the outer loop and estimates message bits at the end of I-th iteration.

One of the important parameters of polar codes under any decoding scheme is the rate of channel polarization which describes how fast the capacity of the transformed bit channels approaches 1 and 0, respectively as $N \to \infty$. We refer the interested readers to [6] for further details about this parameter and mention here the advantage of using the SC decoder in place of the SCAN decoder with $I=1$ for AWGN channels. We observe that by clipping the LLRs of already detected bits to $+\infty, -\infty$ we can increase the convergence and polarization rate. Zimmermann et al. [17] observed the same phenomenon in belief propagation decoder for LDPC codes and called it '*belief pushing*'. It is noteworthy here that the SCAN decoder with $I=1$ is different from the SC decoder, because the SC decoder clips the LLRs of already detected bits in the factor graph to either $+\infty$ or $-\infty$, whereas the SCAN decoder uses soft information about these bits. However, both of these decoders do not use information about the bits yet to be detected and are similar in this respect. With this in mind, one can convert the SCAN decoder with $I=1$ into the SC decoder by assigning $B_n(k,0) = \infty \times \text{sgn}(B_n(k,0)+L_n(k,0))$ as we calculate $\{L_n(k,0)\}_{k=0}^{N-1}$, where sgn(.) is the sign function. Therefore, we can consider the SC decoder as a particular instance of the more general SCAN decoder. We conclude this section by presenting the following proposition.

Proposition 1: The rate of channel polarization is higher under the SC decoder than the SCAN decoder with $I=1$. The proof is provided in the appendix.

IV. The Memory-Efficient SCAN Decoder

In [8] and [18], a memory-efficient version of the SC decoder has been proposed by modifying L and B memory indexing. The proposed modifications reduced the memory requirement for L and B to $2N-1$ and $4N-2$, respectively. We show that the modification that [8] proposed for L can be directly applied to the SCAN decoder, and the memory requirement for L can be reduced to $2N-1$ from $N(n+1)$. On the other hand, the modification that [8] proposed for B is not directly applicable for the reasons explained later. As one of the contributions of this paper, we propose a partitioning method for B that reduces its memory requirement to $4N-2+Nn/2$ from $N(n+1)$.

We first briefly describe the modification proposed for L and apply it directly to the SCAN decoder. Looking at the factor graph in Fig.2 and description in Section II, it is clear that all the Φ-groups on a single λ depth has the same number of nodes $2^{n-\lambda}$. Let us denote L and B values corresponding to $\Phi_\lambda(\phi)$ as $\mathcal{L}_\lambda(\phi)$ and $\mathcal{B}_\lambda(\phi)$, respectively. As we calculate $\{L_n(i,0)\}_{i=0}^{N-1}$, traversing i in ascending order, we use the Φ-groups at different depths in ascending order as well. With this schedule of LLR update, when we are updating $\mathcal{L}_\lambda(i)$, we do not need any of the $\{\mathcal{L}_\lambda(\phi) : \phi < i\}$. Therefore, we can overwrite the values of previously calculated $\{\mathcal{L}_\lambda(\phi) : \phi < i\}$ and only need $2^{n-\lambda}$ memory locations for a depth λ. Hence, the total number of memory elements required by L is $\sum_{\lambda=0}^{n} N/2^\lambda = 2N-1$. Keeping in view the similarity of LLR updates in both the SC decoder and the SCAN decoder, we propose this modification to the later, and it reduces the memory requirement for L from $N(n+1)$ to $2N-1$. It is noteworthy that this modification is not possible in the similar fashion to the originally proposed belief propagation decoder of [11] because of the so-called 'flooding' nature of LLR update between L and B.

The modification for B in [8] (where B used binary values) that is similar to the modification for L described above, is not applicable to the SCAN decoder, because now, not only do we need to calculate LLRs in B, but we also need to pass them onto the next iteration. Therefore, as [8] suggested for the SC decoder, we cannot overwrite the values of B.

To introduce the modifications to B in the SCAN decoder, we first present the following notation and lemmas. Consider $\mathcal{L}_\lambda(\phi)$ and $\mathcal{L}_\lambda(\delta)$ at any depth $\lambda, \forall\phi \neq \delta$ and $\phi,\delta \in \{0,\ldots,2^\lambda-1\}$. We denote

$$\mathcal{L}_\lambda(\phi) \prec \mathcal{L}_\lambda(\delta)$$

to show that the decoder updates $\mathcal{L}_\lambda(\phi)$ before $\mathcal{L}_\lambda(\delta)$.

Lemma 2: At any depth $\lambda \in \{0,\ldots,n\}$, the decoder updates Φ-groups for both L and B in ascending order from $\phi = 0 \to N-1$, i.e.,

$$\mathcal{L}_\lambda(\phi) \prec \mathcal{L}_\lambda(\phi+1)$$

$$\mathcal{B}_\lambda(\phi) \prec \mathcal{B}_\lambda(\phi+1)$$

for all $\phi \in \{0,\ldots,2^\lambda-2\}$.

Proof: We prove this lemma using mathematical induction. First we note the following trivial cases:

1) The decoder does not update B and L for $\lambda = n$ and $\lambda = 0$, respectively.
2) The decoder trivially updates B in ascending order for $\lambda = 0$, because there is only one Φ-group.

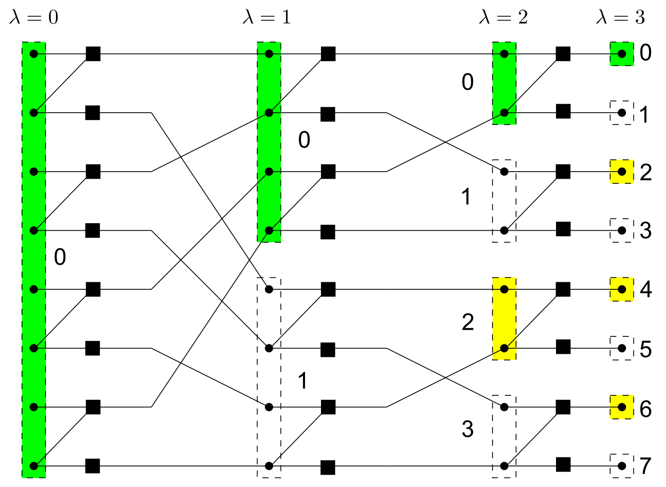


Fig. 2. Memory elements required to store B with corresponding ϕ displayed next to a particular $\mathcal{B}_\lambda(\phi)$. In any iteration, the SCAN decoder does not need $\{\mathcal{B}_\lambda(\phi) : \phi \text{ is even}\}$ for the next iteration, and we can overwrite $\mathcal{B}_\lambda(0)$ (shown with green rectangles) at any depth λ with $\{\mathcal{B}_\lambda(\phi) : \phi \text{ is even}, \phi \neq 0\}$ (shown with yellow rectangles). On the other hand, the SCAN decoder requires $\{\mathcal{B}_\lambda(\phi) : \phi \text{ is odd}\}$ (shown with white rectangles) for processing in the next iteration, and therefore it will keep these memory locations as they are. In this small example, we save five memory elements corresponding to $\mathcal{B}_2(2), \mathcal{B}_3(2), \mathcal{B}_3(4)$ and $\mathcal{B}_3(6)$.

3) The decoder trivially updates L in ascending order for $\lambda = n$ because of the schedule of the decoder on this depth.

First we prove this Lemma for L only. Let us denote $\{\phi_i^\lambda\}_{i=0}^{2^\lambda}$ as the sequence in which we update L at any depth λ. From the schedule of the decoder, we know that $\{\phi_i^n = i\}_{i=0}^{2^n}$. Suppose, that $\{\phi_i^k = i\}_{i=0}^{2^k}$ is true. From (11) and (12), we know that the update in $\mathcal{L}_\lambda(\phi)$ requires the update in $\mathcal{L}_{\lambda-1}(\lfloor\phi/2\rfloor)$. Therefore, $\{\phi_i^{(k-1)} = i\}_{i=0}^{2^{k-1}}$ is also true from the definition of the 'floor' function. We can use the same argument for both the base and induction step of the proof.

Similarly, with (9) and (10), we can prove the same results for B. ■

Algorithm 4: The memory-efficient SCAN decoder

```
Result: Extrinsic LLRs $\{E_0(0,\omega)\}_{\omega=0}^{N-1}$
$\{L_0(0,i)\}_{i=0}^{(N-1)}$ ← LLRs from channel
$\{O_n(i,0)\}_{i\in\mathcal{I}^c, i \text{ is odd}} \leftarrow \infty$
for $i = 1 \rightarrow I$ do
    for $\phi = 0 \rightarrow (N-1)$ do
        updatellrmap(n, $\phi$)
        if $\phi$ is even then
            if $\phi \in \mathcal{I}^c$ then $E_m(\phi,0) \leftarrow \infty$
            else $E_m(\phi,0) \leftarrow 0$
        else updatebitmap(n, $\phi$)
0
```

Lemma 3: At any depth $\lambda \in \{1,\dots,n-1\}$,

$$\mathcal{L}_\lambda(\phi) \prec \mathcal{B}_\lambda(\phi) \prec \mathcal{L}_\lambda(\phi+1), \tag{15}$$

where $\phi \in \{0,\dots,2^\lambda-2\}$.

Proof: Without loss of generality, consider the calculation of $\mathcal{L}_\lambda(\phi)$ and $\mathcal{L}_\lambda(\phi+1)$ for ϕ even, and $\lambda \in \{2,\dots,n\}$. From

Algorithm 5: updatebitmap(λ,ϕ)

```
$\psi \leftarrow \lfloor \frac{\phi}{2} \rfloor$
if $\phi$ is odd then
    for $\omega = 0 \rightarrow (2^{n-\lambda}-1)$ do
        if $\psi$ is even then
            $E_{\lambda-1}(\psi,2\omega) \leftarrow E_\lambda(\phi-1,\omega) \boxplus [O_\lambda(\phi,\omega) + L_{\lambda-1}(\psi,2\omega+1)]$
            $E_{\lambda-1}(\psi,2\omega+1) \leftarrow O_\lambda(\phi,\omega) + E_\lambda(\phi-1,\omega) \boxplus L_{\lambda-1}(\psi,2\omega)$
        else
            $O_{\lambda-1}(\psi,2\omega) \leftarrow E_\lambda(\phi-1,\omega) \boxplus [O_\lambda(\phi,\omega) + L_{\lambda-1}(\psi,2\omega+1)]$
            $O_{\lambda-1}(\psi,2\omega+1) \leftarrow O_\lambda(\phi,\omega) + E_\lambda(\phi-1,\omega) \boxplus L_{\lambda-1}(\psi,2\omega)$
    if $\psi$ is odd then updatebitmap($\lambda-1,\psi$)
```

Lemma.2, (11) and (12) we know that

$$\mathcal{L}_{\lambda-1}(\psi) \prec \mathcal{L}_\lambda(\phi) \prec \mathcal{L}_\lambda(\phi+1),$$

where $\psi = \lfloor\phi/2\rfloor$. Also from (9) and (10),

$$\mathcal{L}_\lambda(\phi+1) \prec \mathcal{B}_{\lambda-1}(\psi).$$

Therefore, using these two relationships we get

$$\mathcal{L}_{\lambda-1}(\psi) \prec \mathcal{B}_{\lambda-1}(\psi).$$

Now considering the calculation of $\mathcal{L}_\lambda(\phi+2)$ and $\mathcal{L}_\lambda(\phi+3)$, we get

$$\mathcal{L}_{\lambda-1}(\psi+1) \prec \mathcal{B}_{\lambda-1}(\psi+1).$$

From Lemma.2, we know that at any λ, the decoder updates both L and B in ascending order, we conclude

$$\mathcal{L}_{\lambda-1}(\psi) \prec \mathcal{B}_{\lambda-1}(\psi) \prec \mathcal{L}_{\lambda-1}(\psi+1),$$

for all $\lambda \in \{2,\dots,n\}$, and $\psi \in \{0,\dots,2^{\lambda-1}-2\}$. We complete the proof by changing variables. ■

Theorem 1: In any iteration i and for any depth λ, the SCAN decoder requires only $\{\mathcal{B}_\lambda(\phi) : \phi \text{ is odd}\}$ from iteration $(i-1)$ to update L.

Proof: Consider (11) for the iteration i. From Lemma.3, we know that

$$\mathcal{L}_\lambda(\phi) \prec \mathcal{B}_\lambda(\phi).$$

Therefore, when the decoder is updating $\mathcal{L}_\lambda(\phi)$, $\mathcal{B}_\lambda(\phi+1)$ is holding the value from iteration $(i-1)$. Since it is true for ϕ even only, $(\phi+1)$ is odd and we use $\{\mathcal{B}_\lambda(i) : i \text{ is odd}\}$ from $(i-1)$. Similarly, (12) shows that to update $\mathcal{L}_\lambda(\phi)$ for odd ϕ, we need $\mathcal{B}_\lambda(\phi-1)$ that, by Lemma 3, the decoder has already updated.Therefore, $\mathcal{B}_\lambda(\phi-1)$ contains the values calculated in the current iteration i. ■

Suppose, we reserve two separate memory locations for B: one to hold $\{\mathcal{B}_\lambda(\phi) : \phi \text{ is even}\}$, namely E and one for $\{\mathcal{B}_\lambda(\phi) : \phi \text{ is odd}\}$, namely O. From Theorem 1, we conclude that we only need to keep O for the next iteration with only

Algorithm 6: updatellrmap(λ, ϕ)

```
if $\lambda = 0$ then return $\psi \leftarrow \lfloor \frac{\phi}{2} \rfloor$
if $\phi$ is even then updatellrmap($\lambda - 1$, $\psi$) for
$\omega = 0 \rightarrow (2^{n-\lambda} - 1)$ do
    if $\phi$ is even then
        $L_\lambda(\phi,\omega) \leftarrow L_{\lambda-1}(\psi,2\omega) \boxplus [L_{\lambda-1}(\psi,2\omega+1) + E_\lambda(\phi+1,\omega)]$
    else
        $L_\lambda(\phi,\omega) \leftarrow L_{\lambda-1}(\psi,2\omega+1) + L_{\lambda-1}(\psi,2\omega) \boxplus O_\lambda(\phi-1,\omega)$
```

$N/2$ elements at a depth λ. In contrast, the decoder will use E in the current iteration only, and therefore the decoder can use the same space $\mathcal{B}_\lambda(0)$ for all $\{\mathcal{B}_\lambda(\phi), \phi \text{ is even}\}$ at a depth λ by overwriting it. The number of memory elements required for E is exactly the same as required for L, i.e., $(2N-1)$. The decoder also needs to specify the indexing of both E and O. As noted in [8], for E ϕ does not convey any information because the decoder writes all the values to the same location at a depth λ, similar to L, and therefore it can use the same indexing for both E and L. One such memory indexing function is

$$f(\lambda,\omega) = \omega + 2^{(n+1)} - 2^{(n+1-\lambda)}. \quad (16)$$

Since O is used only for odd values of ϕ, we can convert these odd values into the natural numbers by a simple transformation and then use it to index O. One such indexing function is

$$g(\lambda,\phi,\omega) = \omega + (\phi-1)2^{(n-\lambda-1)} + (\lambda-1)\,2^{(n-1)}. \quad (17)$$

Fig. 2 presents a small example of a rate $1/2$ polar code. In this example, the SCAN decoder reuses $\mathcal{B}_2(0)$ and $\mathcal{B}_3(0)$ (shown with green rectangles) by overwriting them with the values of $\mathcal{B}_2(2), \mathcal{B}_3(2), \mathcal{B}_3(4)$ and $\mathcal{B}_3(6)$ and does not need extra memory for them. On the other hand, the SCAN decoder keeps the values for $\{\mathcal{B}_\lambda(\phi), \forall\lambda, \phi \text{ is odd}\}$ as they are required for the next iteration.

We summarize the details of the proposed low-complexity SCAN decoder in Algorithm 4, 6 and 5. Algorithm 4 is the top-level wrapper for the SCAN decoder, similar to Algorithm 1. The SCAN decoder successively calls Algorithm 6 and 5 as it traverses all the uncoded bits from $i = 0$ to $N-1$. Algorithm 5 updates the two portions of B using L as prior information: E for the groups with ϕ even and O for the groups with ϕ odd, whereas Algorithm 6 updates L using E and O as prior information. It is noteworthy that in all the algorithms we have indexed E and O using (16) and (17), respectively.

V. A Comparison with the BP decoder

In this section, we compare the BP decoder for polar codes with the SCAN decoder. The prime difference between the two decoders lies in the schedule of LLR updates. As explained later, the better dissemination of information in the SCAN decoder results in a more rapid convergence compared to the BP decoder. We explain this difference of the schedule between the two decoders using Fig. 2.

Consider the operation of the BP decoder on the factor graph shown in Fig. 2. Just like the SCAN decoder, the BP decoder also uses two memory locations L and B. The decoder starts by updating LLRs $L_3(0,0), L_3(1,0), B_2(0,0)$ and $B_2(0,1)$ using $\mathcal{L}_2(0), B_3(0,0)$ and $B_3(1,0)$. In this way, the decoder updates the LLRs corresponding to the top-right protograph and then repeats the same process for all the four protographs under $\lambda = 2$. After the updates in the protographs under $\lambda = 2$, the decoder updates the four protographs under $\lambda = 1$ and then $\lambda = 0$ completing its first iteration. Following points are noteworthy in this schedule :

1) The decoder updates L and B on a protograph-by-protograph basis.
2) In any iteration to update any LLR in both L and B, the decoder uses the values in B that are updated in the current iteration and the values in L updated in the previous iteration (or in the case of first iteration, the initialized values of L).
3) When the BP decoder is updating the LLRs under $\lambda = 0$ at the end of the first iteration, the information received from the channel in $\mathcal{L}_0(0)$ moves from the protographs under $\lambda = 0$ to the protographs under $\lambda = 1$. Therefore, in the first iteration the information from the fixed bits travels from the right-most end of the factor graph to the left-most end, but the information received from the channel moves only to the neighboring protographs, i.e., the protographs under $\lambda = 1$. Following the same procedure, we can show that in every iteration, the information about the fixed bits traverses the whole factor graph, whereas the information received from the channel moves to the neighboring protographs only, and it requires n iterations to reach the right-most end of the factor graph.

The SCAN decoder updates $\mathcal{L}_1(0)$, $\mathcal{L}_2(0)$, $\mathcal{L}_3(0)$ and $\mathcal{L}_3(1)$ in this order. After the update in $\mathcal{L}_3(1)$, the SCAN decoder updates $\mathcal{B}_2(0)$ using $\mathcal{L}_2(0)$ that has just been updated. In this way, as the SCAN decoder updates $\mathcal{L}_3(i)$ from $i = 0$ to 7, it populates B using the updated values in L. At the end of the first iteration, the information received from the channel moves from left end of the factor graph to the right while the information about the fixed bits move from the right end to the left. Following points are noteworthy in this schedule:

1) The decoder does not update L and B on protograph-by-protograph basis; instead it is a node-by-node basis update schedule except in the protographs under $\lambda = 2$. For example, the SCAN decoder first updates $\mathcal{L}_1(0)$, $\mathcal{L}_2(0)$, $\mathcal{L}_3(0)$ that are the updates corresponding to the top-right node of the protographs involved.
2) In any iteration to update any LLR in B, the SCAN decoder uses B as well as L updated in the current iteration. To update any LLR in L, the decoder uses L and $\{\mathcal{B}_\lambda(\phi) : \phi \text{ even}\}$ updated in the current iteration while $\{\mathcal{B}_\lambda(\phi) : \phi \text{ odd}\}$ updated in the previous iteration.
3) In any iteration, the information about both the fixed bits in $\{\mathcal{B}_3(i)\}_{i=0}^{7}$ and the information received from the channel in $\mathcal{L}_0(0)$ traverse the entire factor graph.

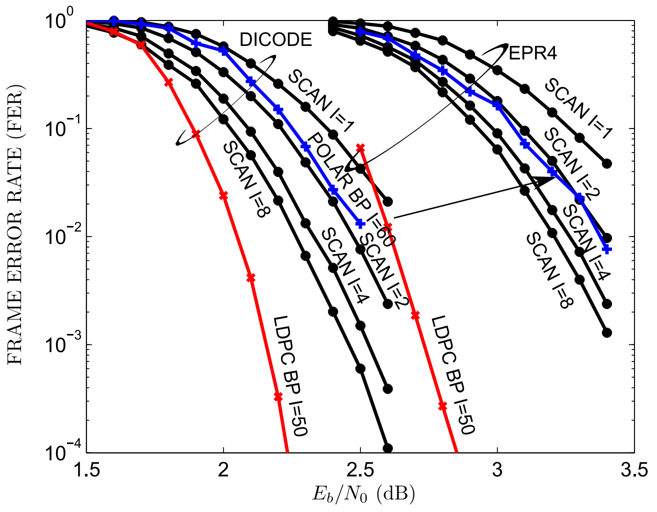


Fig. 3. FER performance of the SCAN decoder in partial response channels for $K = 4096$ and $N = 8192$. We have optimized the polar code for $E_b/N_0 = 1.4$ dB (dicode) and $E_b/N_0 = 1.55$ dB (EPR4) using the method of [15].

As described above, the BP decoder needs at least n iterations to disperse the information contained in $\{\mathcal{B}_n(i)\}_{i=0}^{(N-1)}$ and $\mathcal{L}_0(0)$ in the entire factor graph, whereas the SCAN decoder achieves this with only one iteration. In this way, the SCAN decoder achieves a faster convergence by better disseminating the information in the factor graph than the BP decoder as pointed out by the last two points of the schedules in both the decoders.

VI. Complexity Analysis and Simulation Results

A. AWGN

To demonstrate the improved performance of our algorithm, we have simulated the SCAN decoder for a block length of $N = 32768$ and dimension $K = 16,384$ on the AWGN channel. We have simulated a maximum of 10^6 frames, terminating the simulation if 100 or more frames are found in error. Fig. 4 shows the simulation results, showing that the SCAN decoder outperforms the SC decoder both in FER and BER performance with only two and one iteration, respectively. Additionally, the SCAN decoder exhibits larger gain in BER performance as compared to FER performance.

B. Partial Response Channels

Fig. 3 shows the performance of the proposed decoder on the dicode channel with $N = 8192$ and dimension $K = 4096$ under turbo equalization architecture [7]. The SCAN decoder with only two iterations outperforms the BP decoder with 60 iterations on the dicode channel. Specifically, on the dicode channel, the SCAN decoder's processing and memory requirements are 4% and 43% of those required by the BP decoder, respectively. With the further increase in number of iterations, the performance improves with increase in the computation complexity. We also compare the polar code's performance with the SCAN decoder to that of an irregular LDPC code of the variable node distribution $\lambda(x) = 0.2184x + 0.163x^2 + 0.6186x^3$, check node distribution $\rho(x) = 0.6821x^4 + 0.3173x^5 + 0.0006x^6$, average column weight $d_v = 3.4$ and row weight $d_c = 5.31$ decoded using the BP algorithm and constructed using [19], [20] and [21]. The performance difference between this LDPC code with the BP algorithm and the polar code with the SCAN decoder (using four iterations) is approximately 0.3 dB for FER = 10^{-3} on a dicode channel. The performance loss in the case of EPR4 channel is larger than that in the case of a dicode channel. This performance difference between the two families of codes under message passing decoding is expected, because polar codes are structured codes and this LDPC code is a random one. The structure in LDPC codes also, in general, results in worse performance and increases the complexity of the decoder [22]. Furthermore, it has been shown that polar codes outperform LDPC codes if concatenated with very simple codes [8], [9] in AWGN channel. In this respect, the SCAN decoder can have potential applications for turbo decoding of concatenated codes because of their ability to provide soft outputs needed.

TABLE I
Complexity Comparison of Different Decoders

	Complexity/Iteration	
Operation	LDPC BP	Polar SCAN/BP
Table Lookups	$(N-K)(d_c+1)$	$6Nn$
Multiplications	$(N-K)(d_c-1)$	$2Nn$
Divisions	$(N-K)d_c$	0
Additions/Subtractions	$2Nd_v$	$2Nn$
Total Operations	$5Nd_v$	$10Nn$

C. Complexity

Table I compares the complexity of different decoders for LDPC and polar codes. We have used the complexity analysis for LDPC codes given in [23], where one iteration consists of variable to check and then check to variable message passing. We have further assumed that the decoder uses table lookup method to calculate both $\tanh(.)$ and $\tanh^{-1}(.)$. The number of operations required for the SCAN decoder with four iterations in the dicode channel is approximately equal to 70% of that required for the BP decoder for the LDPC code with 50 iterations, as shown in Fig. 3. This highlights the complexity reduction relative to this LDPC code, along with other benefits of polar codes with some (about 0.3 dB at FER= 10^{-3}) loss in performance.

Fig. 5 shows how the normalized memory requirement decreases with the increase in n. The BP decoder uses $N(n+1)$ floating-point elements for each of L and B. The SCAN decoder uses $2N-1$ floating-point elements for L and $4N-2+Nn/2$ floating-point elements for B. For the complete operation of the SCAN decoder, we also need another boolean memory of size N to hold the information about the set $\mathcal{I}$. As a numerical example, Fig.5 shows that the memory required by the SCAN decoder at two practical frame lengths of 4096 and 32768 is 43% and 39% of that required by the BP decoder, respectively.

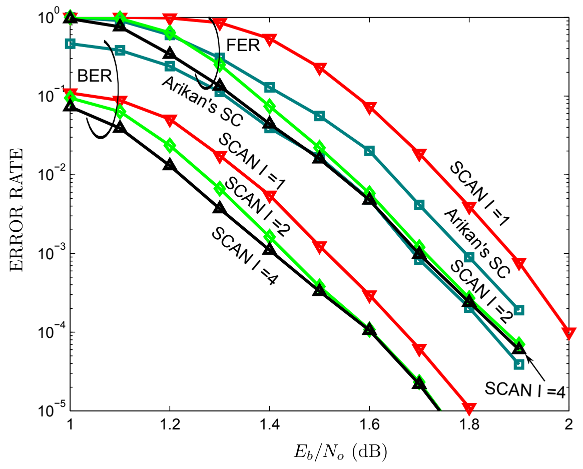


Fig. 4. FER performance of the SCAN decoder in AWGN channel for $N = 32768$. We have optimized the polar code for $E_b/N_0 = 2.35$ dB using the method of [15].

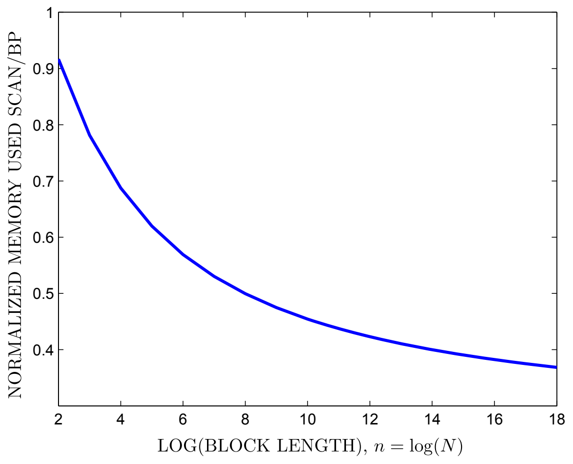


Fig. 5. Memory efficiency improves with increasing block length.

VII. Conclusion and Future Directions

We have proposed SCAN, a soft-output decoder for polar codes that offers good performance, low computational complexity and low memory requirements. We have shown that the SCAN decoder's computational complexity with two iterations is approximately 4% that of the BP decoder with 60 iterations on a dicode channel with comparable performance. The SCAN decoder's performance improves with the increase in the number of iterations. Furthermore, we have proved that the SCAN requires $Nn/2$ (unlike $N(n+1)$ for the BP decoder) memory elements to pass from one iteration to the next. Using this fact, we have proposed a memory-splitting method in which we keep one portion of the memory needed for the next iterations as it is and optimizes the other one that we use in the current iteration. With our proposed decoder, the memory required by the SCAN decoder is approximately 39% of that required by the BP decoder at a block length $N = 32768$ in one example. We have performed Monte Carlo simulations on the AWGN channel as well as partial response channels to demonstrate the functionality of the algorithms. With this three facet (complexity, performance and memory) improvement, the SCAN decoder stands out as a promising soft-output decoder for polar codes. Our work is a first step towards the incorporation of polar codes in magnetic recording channel. Further research is needed to produce length-compatible polar codes and their high-throughput decoders.

Appendix

Proof of Lemma 1:

$$\begin{aligned} W^-(y_0, y_1, z_1|u_0) &= \sum_{u_1} W_2(y_0, y_1, z_1, u_1|u_0) \\ &= \frac{1}{2}\sum_{u_1} W(y_0|u_0 \oplus u_1)W(y_1|u_1)P(z_1|u_1). \end{aligned} \tag{18}$$

$$\begin{aligned} W^+(y_0, y_1, z_0|u_1) &= \sum_{u_0} W_2(y_0, y_1, z_0, u_0|u_1) \\ &= \frac{1}{2}W(y_1|u_1)\sum_{u_0} W(y_0|u_0 \oplus u_1)P(z_0|u_0), \end{aligned} \tag{19}$$

where we have used the fact that both the bits u_0, u_1 are equally likely to be 0 or 1. Using (18) and (19) with the definition of an LLR, we get (13) and (14). ■

Proof of Proposition 1: Consider the problem setup for (18) and (19). Recall for an SC decoder, we have from [24]

$$\begin{aligned} &Z(W_{SC}^+) = Z(W)^2, \\ &Z(W)\sqrt{2 - Z(W)^2} \le Z(W_{SC}^-) \le 2Z(W) - Z(W)^2, \end{aligned}$$

where $Z(W)$ is Bhattacharrya parameter of the DMC W defined as

$$Z(W) \triangleq \sum_y \sqrt{W(y|0)W(y|1)}. \tag{20}$$

Since, for the SCAN decoder with $I = 1$, the computation for the check-node doesn't change, the relationships for $Z(W^-)$ as described above hold. Therefore, we only need to prove $Z(W^+) \ge Z(W)^2$.

$$\begin{aligned} Z(W^+) &= \sum_{y_0,y_1,z_0} \sqrt{W^+(y_0, y_1, z_0|0)W^+(y_0, y_1, z_0|1)} \\ &= \frac{1}{2}Z(W) \times A(W, P), \end{aligned} \tag{21}$$

where

$$\begin{aligned} A(W, P) = \sum_{y_0,z_0} &\left(\sum_{u_0} W(y_0|u_0)P(z_0|u_0)\right) \\ &\times \left(\sum_{u_0'} W(y_0|u_0' \oplus 1)P(z_0|u_0')\right). \end{aligned}$$

From Lemma 3.15 in [24], we have

$$A(W, P) \ge 2\sqrt{Z(W)^2 + Z(P)^2 - Z(W)^2 Z(P)^2}. \tag{22}$$

Using (22) in (21), we get

$$\begin{aligned} Z(W^+) &= Z(W)^2\sqrt{1+Z(P)^2\left(\frac{1}{Z(W)^2}-1\right)}, \\ &\geq Z(W)^2 \end{aligned}$$

as by definition $0 \leq Z(P), Z(W) \leq 1$. ■

References

[1] H. Zhong, W. Xu, N. Xie, and T. Zhang, “Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording,” *IEEE Trans. Magn.*, vol. 43, no. 12, pp. 4117–4122, 2007.

[2] R. M. Tanner, D. Sridhara, and T. Fuja, “A class of group-structured LDPC codes,” *Proc. of ICSTA*, 2001.

[3] D. J. MacKay and R. M. Neal, “Near shannon limit performance of low density parity check codes,” *Electronics Letters*, vol. 32, no. 18, p. 1645, 1996.

[4] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 2012.

[5] *Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems*, IEEE Std. 802.16, 2009.

[6] E. Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[7] C. Douillard, M. Jzquel, C. Berrou, D. Electronique, A. Picart, P. Didier, and A. Glavieux, “Iterative correction of intersymbol interference: Turbo-equalization,” *European Trans. Telecommun.*, vol. 6, no. 5, pp. 507–511, 1995.

[8] I. Tal and A. Vardy, “List decoding of polar codes,” in *Proc. IEEE Int. Symp. Inform. Theory*, Aug. 2011, pp. 1–5.

[9] B. Li, H. Shen, and D. Tse, “An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check,” *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044 –2047, Dec. 2012.

[10] K. Niu and K. Chen, “Stack decoding of polar codes,” *Electronics Letters*, vol. 48, no. 12, pp. 695–697, Jul. 2012.

[11] E. Arkan, “A performance comparison of polar codes and Reed-Muller codes,” *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 447 –449, Jun. 2008.

[12] N. Hussami, S. Korada, and R. Urbanke, “Performance of polar codes for channel and source coding,” in *Proc. IEEE Int. Symp. Inform. Theory*, 2009, pp. 1488–1492.

[13] I. Tal and A. Vardy, “How to construct polar codes,” *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[14] R. Mori and T. Tanaka, “Performance of polar codes with the construction using density evolution,” *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, Jul. 2009.

[15] P. Trifonov and P. Semenov, “Generalized concatenated codes based on polar codes,” in *8th Int. Symp. Wireless Communication Systems*, Nov. 2011, pp. 442–446.

[16] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[17] E. Zimmermann and G. Fettweis, “Reduced complexity LDPC decoding using forced convergence,” in *Proc. 7th Int. Symp. Wireless Personal Multimedia Communications*, 2004, p. 15.

[18] C. Leroux, I. Tal, A. Vardy, and W. Gross, “Hardware architectures for successive cancellation decoding of polar codes,” in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2011, pp. 1665–1668.

[19] X.-Y. Hu. Source code for Progressive Edge Growth parity-check matrix construction. [Online]. Available: http://www.cs.toronto.edu/~mackay/PEG_ECC.html

[20] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, “Progressive edge-growth tanner graphs,” in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, 2001, pp. 995–1001.

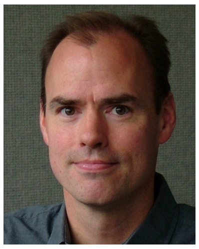
[21] LOPT - online optimisation of LDPC and RA degree distributions. [Online]. Available: http://sonic.newcastle.edu.au/ldpc/lopt/

[22] M. Yang, W. Ryan, and Y. Li, “Design of efficiently encodable moderate-length high-rate irregular LDPC codes,” *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 564–571, 2004.

[23] S. Jeon and B. Kumar, “Performance and complexity of 32 k-bit binary LDPC codes for magnetic recording channels,” *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 2244–2247, 2010.

[24] S. B. Korada, “Polar codes for channel and source coding,” Ph.D. dissertation, EPFL, Lausanne, 2009. [Online]. Available: http://library.epfl.ch/theses/?nr=4461

Ubaid U. Fayyaz Ubaid Fayyaz received the B.S. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan in 2005, and the M.S. degree in electrical engineering from the Georgia Institute of Technology, Georgia USA, in 2013. He is currently pursuing his Ph.D. degree in electrical engineering at the Georgia Institute of Technology Atlanta, Georgia USA. From 2006 to 2009, he worked at the Center for Advance Research in Engineering Islamabad, Pakistan, where he was responsible for the algorithm design and FPGA-based implementations of communication systems. His current research interests include coding theory, information theory and signal processing. He is recipient of the William J. Fulbright, the Water and Power Development Authority Pakistan, and the National Talent scholarships.



John R. Barry Dr. John R. Barry received the B.S. degree summa cum laude from the State University of New York at Buffalo in 1986, and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1987 and 1992, respectively, all in electrical engineering. His doctoral research explored the feasibility of broadband wireless communications using diffuse infrared radiation. Since 1985 he has held engineering positions in the fields of communications and radar systems at Bell Communications Research, IBM T.J. Watson Research Center, Hughes Aircraft Company, and General Dynamics. Currently he is serving as a Guest Editor for a special issue of the IEEE Journal on Selected Areas in Communications. He is a coauthor of Digital Communication, Third Edition, Kluwer, 2004. He is a co-editor of Advanced Optical Wireless Communication Systems, Cambridge University Press, April 2012. He is the author of Wireless Infrared Communications, Kluwer, 1994. He received the 1992 David J. Griep Memorial Prize and the 1993 Eliahu Jury Award from U.C. Berkeley, a 1993 Research Initiation Award from NSF, and a 1993 IBM Faculty Development Award. He is a senior member of IEEE. He is currently serving as Technical Program Chair for IEEE Globecom 2013.