Permuted Successive Cancellation Decoder for Polar Codes

Harish Vangala, Emanuele Viterbo, and Yi Hong,
Dept. of ECSE, Monash University, Melbourne, VIC 3800, Australia.
Email: {harish.vangala, emanuele.viterbo, yi.hong}@monash.edu

***Abstract*—We study a new variant of Arikan's successive cancellation decoder (SCD) for polar codes. We first propose a new decoding algorithm on a new decoder graph, where the various stages of the graph are permuted. We then observe that, even though the usage of the permuted graph doesn't affect the encoder, it can significantly affect the decoding performance of a given polar code. The new *permuted successive cancellation decoder* (PSCD) typically exhibits a performance degradation, since the polar code is optimized for the standard SCD. We then present a new polar code construction rule matched to the PSCD and show in simulations that this can yield BER gains for high code rates. For lower rates we observe that the polar code matched to a given PSCD performs as well as the original polar code with the standard SCD. We also see that a PSCD with a *reversal* permutation can lead to a *natural* decoding order, avoiding the standard *bit-reversal* decoding order in SCD without any loss in performance.**

***Keywords*—Successive cancellation decoder, permuted successive cancellation decoder, decoding order, permuted polar code construction, permutation invariance of the polar encoder.**

I. Introduction

An important property of the first provably capacity achieving polar codes by Arikan [1] is their low complexity decoding of the order of $O(N \log N)$, where N is the block length. For being the first class of provably capacity achieving codes with explicit low complexity recursive structure at both encoder and decoder, polar codes are becoming attractive for practical implementation. Indeed, implementations are reported for polar codes with very high block-lengths up to 2^{17} [2]–[4]

The classic successive cancellation decoder (SCD) for polar codes proposed by Arikan is an important element in proving the capacity theorems for polar codes. In practice, the SCD also exhibits several useful properties such as fixed, deterministic complexity and recursive structure with very good performance. However, in its original form, the decoder is known to have worse performance at finite block-lengths, when compared with the best available LDPC codes [5].

A number of variations to the classic successive cancellation decoder (SCD) are proposed to improve the finite block-length performance of the polar codes [5]–[13]. It is worth noting that the best known decoding algorithms to polar codes continue to exploit the basic successive cancellation algorithm towards better performance, even though many distinct, alternative decoding algorithms are available such as belief propagation [14]–[16]. However, the performance improvements gained by the proposed decoder variations generally come at the cost of a higher complexity compared to the SCD.

This paper was made possible by NPRP grant #NPRP5-597-2-241 from the Qatar National Research Fund (a member of Qatar Foundation).

An important component of the SCD algorithm is its encoding graph, where likelihoods evolve from right to left. The encoder graph is a cascade of $\log_2 N$ stages. Such a graph has the property that any permutation in the order of the $\log_2 N$ stages yields an equivalent encoder. We later came to know that such property was already observed in an example of $N = 8$ in [17], although no proof was given. Here, we give a formal proof.

Although the decoder uses a similar graph structure, the usage of a new graph with permuted stages yields different decoding behavior. The effect of these permutations on the decoder performance has not been studied in the literature. In this paper we study this new variant of SCD, which we call a *permuted successive cancellation decoder* (PSCD), along with its performance and possible enhancements.

The main contributions of this paper are listed below, in the order of their mentioning in this paper.

- We formally prove that an arbitrary permutation in the encoder graph has no effect on the code, supporting the conjecture in [17].
- We show that an arbitrary permutation in the decoder graph may degrade the performance.
- We show that the performance loss is due to the mismatch of the decoder to the original polar code construction.
- We present a new construction rule of polar codes matched to the PSCD, which assures the capacity achieving property (see Sec. III).
- We show by simulation that when a polar code is matched to the PSCD, then its performance will be at least as good as the original polar code with the standard SCD and further gains can be achieved for high rate codes.
- We show that, a PSCD with *reversal* permutation order in the stages of the decoding graph, enables to avoid the bit-reversal permutation involved in classic SCD, resulting in a natural *decoding order* of bits.

The rest of the paper is organized as follows. In the Sec. II, we briefly recall the basics of polar codes. Next, we propose our new permuted polar codes in Sec. III. Sec. IV provides the simulation results and the conclusions are in Sec. V.

II. Polar Coding

A *polar code* is completely specified by the four-tuple $(N, K, \mathcal{F}, \mathbf{v}_{\mathcal{F}})$, where N is the code length in bits (or *block-length*), K is the number of information bits encoded per codeword (or *code dimension*), $\mathbf{v}_{\mathcal{F}}$ is the binary vector of length $N - K$ (*frozen bits*) and $\mathcal{F}$ is a subset of $N - K$

indices from $\{0,1,\ldots,N-1\}$ (*frozen bit locations*).

Given any subset of indices $\mathcal{I}$ from a vector $\mathbf{x}$, we denote the corresponding sub-vector as $\mathbf{x}_{\mathcal{I}}$. Given any subset of column indices $\mathcal{I}$ from a matrix $\mathbf{A}$, we denote the corresponding sub-matrix $\mathbf{A}_{\mathcal{I}}$.

For a $(N,K,\mathcal{F},\mathbf{v}_{\mathcal{F}})$ polar code we describe below the encoding operation for a vector of information bits $\mathbf{u}$ of length K. Let $n=\log_2(N)$ and $\mathbf{G}\triangleq\mathbf{F}^{\otimes n}=\mathbf{F}\otimes\cdots\otimes\mathbf{F}$ be the n-fold Kronecker product of $\mathbf{F}\triangleq\left[\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right]$.

Then, a codeword is generated as

$$\mathbf{x}=\mathbf{G}_{\mathcal{F}^c}\ \mathbf{u}+\mathbf{G}_{\mathcal{F}}\mathbf{v}_{\mathcal{F}}, \tag{1}$$

where $\mathcal{F}^c\triangleq\{0,1,\ldots,N-1\}\backslash\mathcal{F}$ corresponds to the non-frozen bit indices. Alternatively,

$$\mathbf{x}=\mathbf{G}\ \mathbf{d} \tag{2}$$

where $\mathbf{d}\in\{0,1\}^N$ such that $\mathbf{d}_{\mathcal{F}}=\mathbf{v}_{\mathcal{F}}$ and $\mathbf{d}_{\mathcal{F}^c}=\mathbf{u}$.

Note that the second part of (1) is a constant. Typically, the *frozen bits* $\mathbf{v}_{\mathcal{F}}$ are set to zero and we follow this convention throughout the paper.

An efficient implementation of (2) proposed by Arikan, is shown in Fig. 1. Note that in general, there will be n stages.

Construction of Polar Codes – The choice of the set $\mathcal{F}$ is an important step in polar coding often referred to as *polar code construction*. A significant amount of literature is devoted to this operation [1], [18]–[22]. The original algorithm, proposed in [18] and improved in [21], is based on the Bhattacharyya bound approximation. Later proposed algorithms improve on this approximation at the cost of higher complexity. For simplicity, we use the original polar code construction algorithm, which is given by the following recursion [18] for $j=0,\ldots,n-1$ with initial $z_{0,0}=0.5$ (or chosen from [21]),

$$z_{j+1,i}=\begin{cases}2z_{j,i}-z_{j,i}^2\ , & \text{if } 0\le i<2^j\\ z_{j,i-2^j}^2\ , & \text{if } 2^j\le i<2^{j+1}\end{cases} \tag{3}$$

The indices of the highest $N-K$ values in the set of N final stage values $\{z_{n,i}: i=0,1,\ldots,N-1\}$, will be taken as the set $\mathcal{F}$. This algorithm is an evolution from right to left of the Bhattacharyya parameters of channels (see Fig. 1).

Later in this paper, we propose an efficient implementation of the above algorithm for a more general case (Algorithm 2).

Successive Cancellation Decoder (SCD) – The SCD algorithm [1] follows the same encoder diagram in Fig. 1. The likelihood ratio LR (or *likelihood* in short) values evolve in the reverse direction from right-to-left, as explained in [1].

At every stage in Fig. 1, each EXOR adder forms a unit circuit connecting an upper branch to a lower branch through a vertical connection, as labeled. Initially, the likelihoods are computed from the channel observations at the right end of the circuit, and they evolve through the unit circuits from right to left on the graph. Once a likelihood evolves to the left end of the circuit a decision is made and it is propagated from left to right to update the bits on the rest of the graph. Specifically,

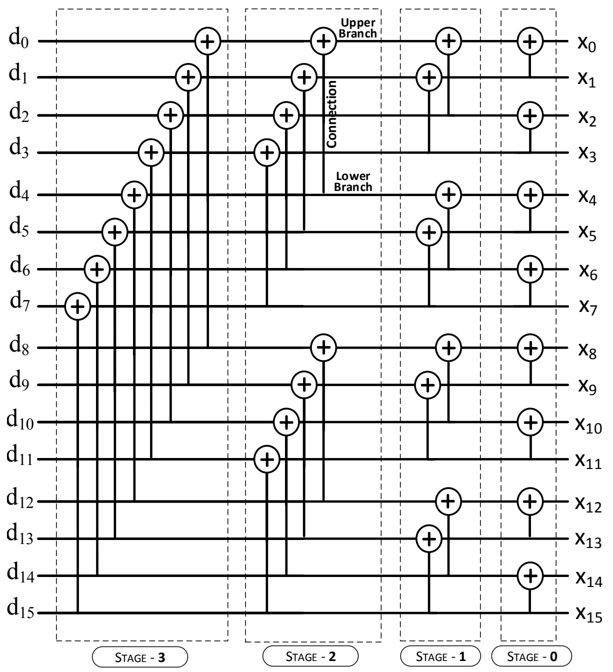

Fig. 1. Arikan's $O(N\log_2 N)$ complexity encoder graph of (2) at $N=16$

in each unit circuit, an upper branch likelihood is calculated first while a lower branch likelihood is calculated only after a decision is available on the upper branch. The details about the likelihood equations update are available in [1]. We remark the special *decoding order* of bits, namely the *bit-reversal* order, which is unique for this decoder structure.

A full implementation of the above standard SCD is given by Algorithm 1 and the two recursive functions **UpdateL** and **UpdateB**, which are given in the next page. Algorithm 1 is the main module calling the recursive functions **UpdateL** and **UpdateB**. The functions are compactly presented with a recursive implementations, but can be easily unfolded into non-recursive form using standard techniques.

The matrices $\mathbf{B}$ and $\mathbf{L}$ denote the matrices of bits and likelihood ratios, respectively. These are accessible to all three modules and make up the $O(N\log N)$ space complexity of the decoder. The function $\mathbf{bitreversal}(i)$ represents the new index obtained by the well-known bit reversal permutation of the n bit binary representation of index i.

III. Permuted Polar Coding

We define a *permutation in SCD* as the rearrangement of the stages $\{0,1,\ldots,n-1\}$ in Fig. 1. The new decoder is called a *permuted successive cancellation decoder* (PSCD).

A PSCD is completely specified by the permutation vector $\underline{\pi}\triangleq[\pi_0,\pi_1,\ldots,\pi_{n-1}]$, where decoding stages $\pi_0,\pi_1\ldots$ appear from left to right. For example, $\underline{\pi}=[n-1,\ldots,1,0]$ denotes Arikan's classic SCD (Fig. 1) and $\underline{\pi}=[0,1,\ldots,n-1]$ denotes the PSCD with fully reversed order of stages.

In the next two subsections, we see the effect of a $\underline{\pi}$ -permuted graph on the encoding and decoding of polar codes.

Algorithm 1 : Main Module of SCD Algorithm

```
INPUT  : $(N, K, \mathcal{F}, \mathbf{v}_{\mathcal{F}} = 0)$ and received $\mathbf{y}$.
OUTPUT: $\hat{\mathbf{u}}$, the estimation of transmitted $\mathbf{u}$.
1:  Allocate and make visible to the other functions $N \times (n+1)$
    matrices $\mathbf{B}, \mathbf{L}$ and set them to NaN.
2:  $n = \log_2 N$
3:  Initialize last column $\mathbf{L}[:][n]$ =Pr$(\mathbf{y}|0)$/Pr$(\mathbf{y}|1)$ with likelihoods
    from channel observations $\mathbf{y}$
4:  for $i = 0, 1, \ldots, N-1$ do
5:      $l = \mathbf{bitreversal}(i)$
6:      $\mathbf{UpdateL}(l, 0)$                ▷ Update $\mathbf{L}[l][0]$ and $\mathbf{L}$
7:      if $l \in \mathcal{F}$ then
8:          $\mathbf{B}[l][0] = 0$
9:      else
10:         $\mathbf{B}[l][0] = \begin{cases} 0, & \text{if } \mathbf{L}[l][0] \geq 1 \\ 1, & \text{else} \end{cases}$
11:     end
12:     $\mathbf{UpdateB}(l, 0)$            ▷ Broadcast bit-$l$ & update $\mathbf{B}$
13: end
14: $\mathbf{d} = \mathbf{B}[:][0]$                   ▷ First column of matrix $\mathbf{B}$
15: $\hat{\mathbf{u}} = \mathbf{d}_{\mathcal{F}^c}$                          ▷ Information bits
```

Function UpdateL(i, j) ▷ Recursive L.R. computation

```
INPUT  : Element indices $i, j$
OUTPUT: Recursively updated matrix $\mathbf{L}$
1:  $s = 2^{n-j}$
2:  $l = (i \bmod s)$
3:  if $l < s/2$ then                          ▷ Upper branch
4:      if $(\mathbf{L}[i][j+1] = \text{NaN})$ then
5:          $\mathbf{UpdateL}(i, j+1)$; end;
6:      if $(\mathbf{L}[i+s/2][j+1] = \text{NaN})$ then
7:          $\mathbf{UpdateL}(i+s/2, j+1)$; end;
8:      $\mathbf{L}[i][j] = \frac{\mathbf{L}[i][j+1]\mathbf{L}[i+s/2][j+1]+1}{\mathbf{L}[i][j+1]+\mathbf{L}[i+s/2][j+1]}$
9:  else                                       ▷ Lower branch
10:     if $\mathbf{B}[i-s/2][j] = 0$ then
11:         $\mathbf{L}[i][j] = \mathbf{L}[i][j+1]\mathbf{L}[i-s/2][j+1]$
12:     else
13:         $\mathbf{L}[i][j] = \frac{\mathbf{L}[i][j+1]}{\mathbf{L}[i-s/2][j+1]}$
14:     end
15: end
```

Effect of Permutation on the Encoder — The encoding in (2) can be written as:

$$\mathbf{G} = \mathbf{F}^{\otimes n} = \prod_{i=0}^{n-1} (\mathbf{I}_{2^{n-1-i}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i}) \tag{4}$$

It can be easily verified that the n stages at the encoder in Fig. 1, are represented by each term in (4). In Appendix we prove that these stages *commute* with each other, i.e.,

$$\begin{aligned}(\mathbf{I}_{2^{n-1-i}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i}) \cdot (\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^j}) \\ = (\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^j}) \cdot (\mathbf{I}_{2^{n-1-i}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i}) \quad \forall i, j\end{aligned} \tag{5}$$

Hence, we conclude that the encoder is permutation-invariant.

Effect of Permutation on the Decoder — Since the encoder is permutation-invariant, it is interesting to know how a decoder would behave w.r.t. a permutation $\underline{\pi}$. It can be easily observed that the likelihoods evolution in the graph from right to left [1] is a nonlinear operation and hence the order of the stages

Function UpdateB(i, j) ▷ Broadcasting of decisions

```
INPUT  : Element indices $i, j$
OUTPUT: Recursively updated matrix $\mathbf{B}$
1:  $s = 2^{n-j}$
2:  $l = (i \bmod s)$
3:  if $l < s/2$ or $j \geq n$ then              ▷ Upper branch
4:      return
5:  else                                       ▷ Lower branch
6:      $\mathbf{B}[i-s/2][j+1] = \mathbf{B}[i][j] \oplus \mathbf{B}[i-s/2][j]$
7:      $\mathbf{B}[i][j+1] = \mathbf{B}[i][j]$
8:      $\mathbf{UpdateB}(i, j+1)$
9:      $\mathbf{UpdateB}(i-s/2, j+1)$
10: end
```

may affect the overall decoder performance. A study of such effects is our main contribution in this paper.

Given a PSCD specified by $\underline{\pi}$, Algorithm 1 must be modified for: (*i*) the appropriate new decoding order replacing bit-reversal decoding order (step-5, Algorithm 1) (*ii*) the evolution of likelihoods (step-6, Algorithm 1) and (*iii*) broadcasting of decisions (step-12, Algorithm 1). Table I below summarizes these modifications.

TABLE I
SUMMARY OF CHANGES IN SCD ALGORITHM FOR PSCD

Module	Line No.	Replace With
Algorithm 1 (SCD)	5	$l = \mathbf{nextindex}(\underline{\pi}, l)$
UpdateL(i,j)	1	$s = 2^{1+\pi_{n-1-j}}$
UpdateB(i,j)	1	$s = 2^{1+\pi_{n-1-j}}$

Note that, the first modification requires a new function rather than bit-reversal function. A naive solution, such as an exhaustive search to find which next channel is computable, will have $O(N)$ additional complexity for each bit. This is avoided by using the function $\mathbf{nextindex}(\underline{\pi}, p)$. The new function $\mathbf{nextindex}(\underline{\pi}, p)$ results in the same complexity as that of a bit-reversal algorithm.

In simulations (Sec. IV) we observe that the performance of such a decoder can be degraded. However, we will show below how we can address this issue.

A polar code is constructed to optimize its performance assuming that the decoder uses Arikan's standard SCD structure. Therefore it may not be *fair* to expect the same performance on a new sub-optimal decoder, for which the code is not matched. This motivates us to design a matching code construction for PSCD. Hence we propose Algorithm 2, an efficient implementation of (3) extended for PSCD, taking the original polar code construction algorithm (3) as a special case, when $\underline{\pi} = [n-1, \ldots, 1, 0]$.

As suggested by a reviewer, the extended code construction (algorithm 2) yields the same estimations of the Bhattacharyya bounds of the bit channels, but in a different order. By similarly freezing the least reliable bits in such new order, we can obtain a similar FER performance, whenever these bounds are tight. More importantly, based on this invariance of bounds to permutation, we can conclude that the PSCD with the matched code construction continues to be capacity achieving.

```
Function nextindex($\underline{\pi}, p$)    ▷Decoding Order
INPUT  : Current index $p$ and permutation $\underline{\pi} = [\pi_0, \ldots, \pi_{n-1}]$
OUTPUT: Next decodable bit index $q$, in the new decoding order
1: if ($p$ == NaN) then $q = 0$ and return.    ▷ First index
2: Let $p \equiv \sum_{t=0}^{n-1} p_t 2^t$ be the binary-representation of $p$
3: $x = \left(\sum_{t=0}^{n-1} p_{\pi_t} 2^t\right)$    ▷ Leftmost stage in $\underline{\pi}$ is LSB
4: $x = x + 1$
5: Let $x \equiv \sum_{t=0}^{n-1} x_t 2^t$
6: $q = \left(\sum_{t=0}^{n-1} x_t 2^{\pi_t}\right)$
```

However, we show in Sec. IV by simulations that the BER of PSCD for matched code (using algorithm 2) can vary due to the new decoding order induced by PSCD, and can outperform the BER of SCD for original polar code at higher rates. Typically, we observed that BER performance of PSCD for matched code is as good as the original.

Note that there are $n!$ alternative permutations for use under PSCD and matched code, having performance as good as SCD and original polar code. Among them, one particular case of interest is the *reversal* permutation i.e., $\underline{\pi} = [0, 1, \ldots, n-1]$, since it has an interesting property that the decoding order becomes equal to the natural order of bits at encoding. This may be of great practical interest, for e.g., to avoid the additional complexity of bit-reversal permutation (or **nextindex**$(\underline{\pi}, p)$ in general), thereby reducing the overall latency of the decoder at *no loss* in performance (see Fig. 3).

IV. Simulations

In this section we consider block-lengths $N = 256, 1024$, and 2048 with rates $R = 0.5, 0.7$, and 0.9, respectively. In the figures, we use abbreviations SCC and PSCC to denote the code constructions for SCD and PSCD, respectively. We consider Binary Input Additive White Gaussian Noise (BI-AWGN) channel in all simulations. Following the discussion in Sec. III, we observed in simulations that FER of PSCD under PSCC is indeed very close to that of a standard polar code under SCD. Due to space limitations, we omit the plots of FER and show only the plots of BER.

Fig. 2 shows the performance comparison of PSCD and SCD where $N = 256, R = 0.5$. We observe that typically there is a performance degradation when PSCD is used instead of SCD for the same polar code. This also says that some permutations can have no significant impact to performance.

Fig. 3 shows that the performance comparison of Arikan's polar code using SCD and PSCD, and the new proposed codes constructed for PSCD at $N = 1024$ and $R = 0.7$. We see that the performance of the new code constructed for PSCD is at least as good as the performance of the classic SCD.

Fig. 4 illustrates the performance of the new code construction using PSCC and Arikan's original polar code using SCD and PSCD at $N = 2048$ and $R = 0.9$. It shows that the potential performance gain can be obtained for some permutations at higher rates, when a PSCD along with a matched

```
Algorithm 2 : Polar Code Construction for PSCD
INPUT  : $N, K, n \triangleq \log_2 N$, initial seed $z_0$ and the permutation
         $\underline{\pi} = [\pi_0, \pi_1, \ldots, \pi_{n-1}]$
OUTPUT: $\mathcal{F} \subset \{0, 1, \ldots, N-1\}$ with $|\mathcal{F}| = N - K$
1: $z \in \mathbb{R}^N$ and $z[i] = z_0 \ \forall i = 0, 1, \ldots, N-1$
2: for $j = 0 : n-1$ do    ▷ for each stage (right-to-left)
3:     $s = 2^{1+\pi_{n-1-j}}$    ▷ the sub-stage size
4:     for $l = 0 : \frac{N}{s} - 1$ do    ▷ For each sub-stage
5:         for $t = 0 : \frac{s}{2} - 1$ do    ▷ For each connection
6:             $T = z[ls + t]$
7:             $z[ls + t] = 2T - T^2$    ▷ Upper channel
8:             $z[ls + s/2 + t] = T^2$    ▷ Lower channel
9:         end
10:    end
11: end
12: $[z, idx] = \mathbf{Sort}(z, \text{'descending'})$ ▷ obtain in $idx$, the indices
      of $z$ vector when sorted in descending order
13: $\mathcal{F} = idx[0 : N-K-1]$    ▷ return first $N - K$ indices
```

code construction is used. These results demonstrate that the permuted polar codes designed for PSCD can offer significant gains over original polar codes at the same complexity.

V. Conclusion

We have proposed a new variant of Arikan's SCD algorithm, which involves reordering of the decoding stages within SCD. We have found that the performance is at least as good as the original decoder once we match the code construction to the PSCD, while the performance is typically degraded when they are not matched. We have seen that the performance gains can be significant at high rates. We also saw how a particular *reversal* permutation can avoid the bitreversal order of decoding and outputs bits in natural order.

Appendix

Proof of encoder's permutation-invariance

Consider the following identity for Kronecker product operation of sets of matrices $\{\mathbf{A}_1, \ldots, \mathbf{A}_n\}$ and $\{\mathbf{B}_1, \ldots, \mathbf{B}_n\}$, each set containing square matrices of the same dimensions.

$$(\mathbf{A}_1 \cdots \mathbf{A}_n) \otimes (\mathbf{B}_1 \cdots \mathbf{B}_n) = (\mathbf{A}_1 \otimes \mathbf{B}_1) \cdots (\mathbf{A}_n \otimes \mathbf{B}_n) \quad (6)$$

Using the above property, we can rewrite the Kronecker products $\mathbf{F}^{\otimes 2}, \mathbf{F}^{\otimes 3}, \ldots$ as follows.

$$\begin{aligned}\mathbf{F} \otimes \mathbf{F} &= (\mathbf{I}_2 \cdot \mathbf{F}) \otimes (\mathbf{F} \cdot \mathbf{I}_2) \\ &= (\mathbf{I}_2 \otimes \mathbf{F}) \cdot (\mathbf{F} \otimes \mathbf{I}_2) \qquad \text{(using (6))}\end{aligned}$$

$$\begin{aligned}\mathbf{F} \otimes \mathbf{F} \otimes \mathbf{F} &= \mathbf{F} \otimes ((\mathbf{I}_2 \otimes \mathbf{F}) \cdot (\mathbf{F} \otimes \mathbf{I}_2)) \qquad \text{(using above)} \\ &= (\mathbf{I}_2 \cdot \mathbf{I}_2 \cdot \mathbf{F}) \otimes ((\mathbf{I}_2 \otimes \mathbf{F}) \cdot (\mathbf{F} \otimes \mathbf{I}_2) \cdot \mathbf{I}_4) \\ &= (\mathbf{I}_4 \otimes \mathbf{F}) \cdot (\mathbf{I}_2 \otimes \mathbf{F} \otimes \mathbf{I}_2) \cdot (\mathbf{F} \otimes \mathbf{I}_4) \quad \text{(using (6))}\end{aligned}$$

By induction we have (see (4)):

$$\mathbf{F}^{\otimes n} = \prod_{i=0}^{n-1} (\mathbf{I}_{2^{n-1-i}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i})$$

where we can easily verify that each stage in Fig. 1 is implemented by one of the product terms.

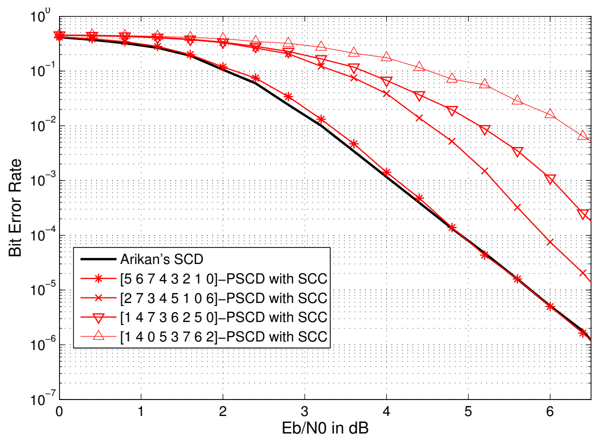

Fig. 2. Performance comparison of PSCD and SCD under original polar construction rule at $N = 256$ and $R = 0.5$ in BI-AWGN channel

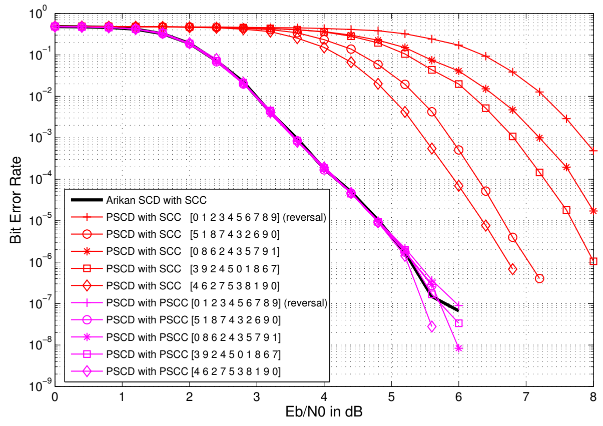

Fig. 3. Performance comparison of SCD under original polar code construction, PSCD under same construction, and PSCD under new code construction at $N = 1024$ and $R = 0.7$ in BI-AWGN channel

Now, without loss of generality, consider any two elements of the above product (4) with distinct indices j, i $(j > i)$.

$$\begin{aligned}
&(\mathbf{I}_{2^{n-1-i}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i}) \cdot (\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^j}) \\
&= \{\mathbf{I}_{2^{n-j}} \otimes (\mathbf{I}_{2^{j-i-1}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i})\} \cdot \{(\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F}) \otimes \mathbf{I}_{2^j}\} \\
&= \{\mathbf{I}_{2^{n-j}} \cdot (\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F})\} \otimes \{(\mathbf{I}_{2^{j-i-1}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i}) \cdot \mathbf{I}_{2^j}\} \\
&\hfill \text{(using (6))} \\
&= \{(\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F}) \cdot \mathbf{I}_{2^{n-j}}\} \otimes \{\mathbf{I}_{2^j} \cdot (\mathbf{I}_{2^{j-i-1}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i})\} \\
&= (\mathbf{I}_{2^{n-1-j}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^j}) \cdot (\mathbf{I}_{2^{n-1-i}} \otimes \mathbf{F} \otimes \mathbf{I}_{2^i}) \quad \text{(using (6))}
\end{aligned}$$

The above proves that any two terms of (4) commute with each other. This implies that any arbitrary permutation in the encoder graph is equivalent to the original graph.

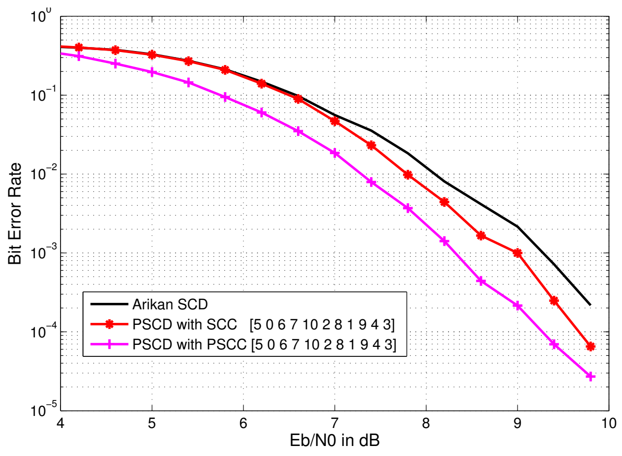

Fig. 4. Performance gain with PSCD under new code construction rule at $N = 2048$ and $R = 0.9$ in BI-AWGN channel

References

[1] 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, July 2009.

[2] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency sc polar decoder architectures," in *International Conference on Communications (ICC)*, Ottawa, ON, June 2012, pp. 3471–3475.

[3] A. Pamuk and E. Arikan, "A two phase successive cancellation decoder architecture for polar codes," in *International Symposium on Information Theory Proceedings (ISIT)*, Istanbul, July 2013, pp. 957–961.

[4] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 289–299, January 2013.

[5] I. Tal and A. Vardy, "List decoding of polar codes," in *International Symposium on Information Theory*, August 2011, pp. 1–5.

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

[7] Z. Huang, C. Diao, and M. Chen, "Multiple candidates successive cancellation decoding of polar codes," in *Int. Conf. on Wireless Comm. and Sig. Proc. (WCSP)*, Huangshan, 2012, pp. 1–4.

[8] K. Niu and K. Chen, "Crc-aided decoding of polarcodes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, October 2012.

[9] ——, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, no. 12, pp. 695–697, June 2012.

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

[11] S. Kahraman, E. Viterbo, and M. E. Celebi, "Folded successive cancellationation decoding of polar codes," in *Australian Communications Theory Workshop (AusCTW), 15th Annual*, 2014, pp. 57–61.

[12] H. Vangala, E. Viterbo, and Y. Hong, "Improved multiple folded successive cancellation decoder for polar codes," in *31st International Union of Radio Science - General Assembly and Scientific Symposium (URSI-GASS)*, Beijing, China, August 2014, (invited paper).

[13] ——, "A new multiple folded successive cancellation decoder for polar codes," in *IEEE Information Theory Workshop (ITW)*, Hobart, Tasmania, Australia, November 2014, (submitted).

[14] N. Goela, S. B. Korada, and M. Gastpar, "On lp decoding of polar codes," in *IEEE Information Theory Workshop(ITW)*, Dublin, September 2010, pp. 1–5.

[15] A. Eslami and H. Pishro-Nik, "On finite-length performance of polar codes: Stopping sets, error floor and concatenated design," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 919–929, March 2013.

[16] S. Kahraman, E. Viterbo, and M. E. Celebi, "Folded tree maximum-likelihood decoder for kronecker product-based codes," in *Allerton Conference on Communication, Control and Computing*, 2013.

[17] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *International Symposium on Information Theory (ISIT)*, Seoul, July 2009, pp. 1488–1492.

[18] E. Arikan, "Performance comparison of polar codes and reed-muller codes," *IEEE Communications Letters*, vol. 12, no. 6, pp. 447–449, 2008.

[19] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *International Symposium on Information Theory (ISIT)*, 2009, pp. 1496–1500.

[20] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, October 2013.

[21] S. Zhao, P. Shi, and B. Wang, "Designs of bhattacharya parameter in the construction of polar codes," in *Int. Conf. on Wireless Comm., Networking and Mob. Comp. (WiCOM)*, Wuhan, Sep. 2011, pp. 1–4.

[22] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, vol. 60, no. 11, pp. 1–7, Nov. 2012.