Fast Successive-Cancellation Decoding of Polar Codes: Identification and Decoding of New Nodes

Muhammad Hanif and Masoud Ardakani, *Senior Member, IEEE*

***Abstract*— The decoding latency of polar codes can be reduced by implementing fast parallel decoders in the last stages of decoding. In this letter, we present five such decoders corresponding to different frozen-bit sequences to improve the decoding speed of polar codes. Implementing them achieves significant latency reduction without tangibly altering the bit-error-rate performance of the code.**

***Index Terms*— Polar codes, maximum a posteriori, maximum likelihood, systematic codes, non-systematic codes.**

I. Introduction

POLAR codes, discovered by Arıkan [1], achieve the symmetric capacity of a binary-input, discrete, memoryless channel with low-complexity encoding and decoding. Currently, they are the only capacity-achieving block codes that have an explicit construction [1], [2]. Due to the afore-mentioned characteristics, they are proposed to be used in the new radio-access technology [3].

A primary concern regarding polar codes is the high decoding latency of the successive-cancellation (SC) decoder [1], [4]–[9]. In particular, the serial nature of decoding and computational complexity of check-node operations significantly reduce the throughput of the SC decoder. As such, researchers have identified some particular cases where the decoding can be parallelized and check-node operations can be reduced or eliminated [5]–[9]. In particular, considering a polar code as a concatenation of smaller constituent codes, [5]–[9] propose fast parallel decoders of the constituent codes corresponding to some particular frozen-bit sequences. Amongst them, [5] introduces a simplified-SC (SSC) decoder that implements low-complexity decoders of rate-zero and rate-one nodes to reduce the decoding latency. The authors in [6] present a fast-SSC decoder that further improves the decoding speed by implementing fast parallel decoders of single-parity-check (SPC) node, repetition (REP) node, and some of their mergers (such as REP-SPC, 0-SPC, and 01 nodes) in the decoding tree of polar codes. Lastly, [8] introduces three more fixed-length node mergers (REP-1, 0-REP-SPC, and 001 nodes) in the decoding tree to improve the decoding speed.

In this work, we identify five other constituent codes corresponding to particular frozen-bit sequences and present their fast decoders. Implementing these decoders further improves the decoding speed of the fast-SSC decoder.

Manuscript received June 5, 2017; revised July 15, 2017; accepted August 11, 2017. Date of publication August 15, 2017; date of current version November 9, 2017. This work was supported by the TELUS Corporation and Natural Sciences and Engineering Research Council of Canada. The associate editor coordinating the review of this letter and approving it for publication was M. Baldi. *(Corresponding author: Muhammad Hanif.)*

The authors are with the Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada (e-mail: mhanif@uvic.ca.)

Digital Object Identifier 10.1109/LCOMM.2017.2740305

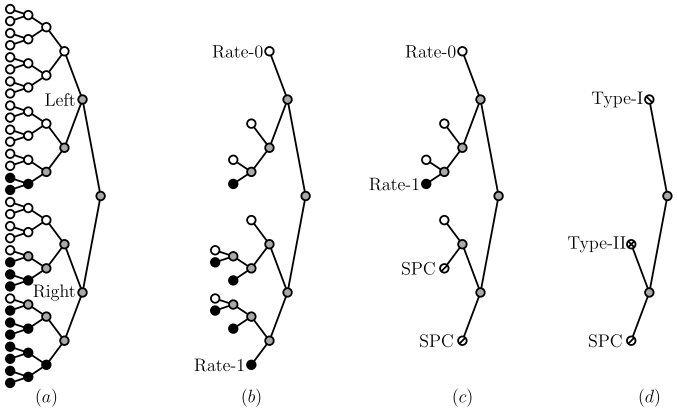

Fig. 1. Decoding trees corresponding to the (a) SC, (b) SSC, (c) fast-SSC, and (d) proposed decoding algorithms.

II. Background

In this section, we provide a brief description of polar codes, primarily to establish some notations and highlight the main contributing factors of high decoding latency of polar codes.

A polar code is constructed based on the symmetric capacity of the polarized bit channels [1]. In particular, the bit channels with the worst capacities are frozen to zero, and the others are used for information transfer. A polar encoder[1] generates its N-bit output, $\mathbf{x}$, such that $\mathbf{x} = \mathbf{u}\mathbf{F}^{\otimes n}$, where $\mathbf{u}$ is an N-bit input vector whose elements are set to zero at frozen bit locations, and $\mathbf{F}^{\otimes n}$ is the nth tensor power of $\mathbf{F}$ defined as

$$\mathbf{F}^{\otimes n} = \begin{bmatrix} \mathbf{F}^{\otimes(n-1)} & \mathbf{O} \\ \mathbf{F}^{\otimes(n-1)} & \mathbf{F}^{\otimes(n-1)} \end{bmatrix}, \tag{1}$$

with $\mathbf{F}^{\otimes 0} = 1$. The information bits, $\mathbf{m}$, appear transparently either in $\mathbf{x}$ or $\mathbf{u}$ depending on whether the polar code is systematic or not, respectively.

A polar code can be represented by a binary code tree (also called a decoding tree), where each node corresponds to a codeword [5], [6]. Fig. 1a depicts a length-32 polar code in its binary-tree representation. Here, white leaf nodes represent the frozen, and black ones represent the information bits.

The SC decoding algorithm operates as follows. The decoding operation starts from the root node, which receives $\mathbf{y} = \begin{bmatrix}\mathbf{y}_0 \ \mathbf{y}_1\end{bmatrix}$, channel log-likelihood ratios (LLR). Here, $\mathbf{y}_0$ and $\mathbf{y}_1$ are the left and right halves of $\mathbf{y}$, respectively. The root node then computes $\widetilde{\mathbf{y}}_0$ whose ith component is computed as

$$\widetilde{y}_{0i} = y_{0i} \circledast y_{1i} = 2\tanh^{-1}\left[\tanh\left(\frac{y_{0i}}{2}\right)\tanh\left(\frac{y_{1i}}{2}\right)\right], \tag{2}$$

where y_{0i} and y_{1i} are the ith elements of $\mathbf{y}_0$ and $\mathbf{y}_1$, respectively.

[1]We consider only non-permuted polar codes for ease of explanation.

The left child of the root node then decodes $\widetilde{\mathbf{y}}_0$ (as explained later) and sends a binary vector $\widetilde{\mathbf{x}}_0$ to the root node. The root node then computes $\widetilde{\mathbf{y}}_1$ and sends it to the right child, where $\widetilde{\mathbf{y}}_1 = (1 - 2\widetilde{\mathbf{x}}_0) \circ \mathbf{y}_0 + \mathbf{y}_1$, and $\circ$ accomplishes entry-wise multiplication. The right child then performs decoding on $\widetilde{\mathbf{y}}_1$ similar to the left child and returns $\widetilde{\mathbf{x}}_1$ to the root node, which computes an estimate of $\mathbf{x}$ as $\widehat{\mathbf{x}} = \left[\widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{x}}_1 \ \widetilde{\mathbf{x}}_1\right]$, where the vector addition is carried out in the binary field.

For non-systematic polar codes, the left child also returns $\widehat{\mathbf{u}}_0$; and the right child, $\widehat{\mathbf{u}}_1$ to the root node. And the root node estimates $\mathbf{u}$ as $\widehat{\mathbf{u}} = \left[\widehat{\mathbf{u}}_0 \ \widehat{\mathbf{u}}_1\right]$.

With the exception of the leaf nodes, each node decodes the received LLRs exactly the same way as the root node does. The leaf nodes, however, either return the frozen bits or hard decision estimates of the information bits to their parents depending on the frozen-bit sequence.

The serial nature of decoding and the computation time of $\circledast$, the check-node operation, are the main contributors of high decoding latency of the SC decoder. To improve the decoding speed, the existing decoders identify and implement fast parallel decoders of R-bit codes corresponding to a frozen-bit sequence, where $R = 2^t$, and t is a positive integer. The SSC decoder [5] implements fast decoders for rate-0 and rate-1 nodes, whereas the fast-SSC decoder [6] also implements fast decoders of the SPC, REP and some merged nodes in the code tree of a polar code. Fig. 1b and Fig. 1c show the pruned decoding trees corresponding to the SSC and fast-SSC decoders, respectively.

In Section III, we identify five more nodes and present fast decoders of their corresponding codes to increase the decoding speed further. We will find the following notations useful.

Let $\{\{N\}\}$ denote the set $\{0, 1, \ldots, N-1\}$. For any $i \in \{\{N\}\}$ and $R = 2^t$, we can write $i = \psi R + [\![i]\!]_t$, where $[\![i]\!]_t$ is the remainder after division of i by R. Further, we define $\tilde{A}_{\psi,t} = \{[\![i]\!]_t : i \in A, \text{ and } i = \psi R + [\![i]\!]_t\}$. Observe that every node of the code tree of a polar code can be uniquely mapped to $\tilde{A}_{\psi,t}$. For example, the leaf nodes correspond to $\tilde{A}_{i,0}$, whereas the root node corresponds to $\tilde{A}_{0,n}$.

With the defined notation, a rate-0 node corresponds to $\tilde{A}_{\psi,t} = \{\}$; rate-1 node, to $\tilde{A}_{\psi,t} = \{\{R\}\}$; REP node, to $\tilde{A}_{\psi,t} = \{R-1\}$; SPC node, to $\tilde{A}_{\psi,t} = \{1, \ldots, R-1\}$; 01 node, to $\tilde{A}_{\psi,2} = \{2, 3\}$; 001 node, to $\tilde{A}_{\psi,3} = \{6, 7\}$; 0-SPC node, to $\tilde{A}_{\psi,3} = \{5, 6, 7\}$; REP-SPC node, to $\tilde{A}_{\psi,3} = \{3, 5, 6, 7\}$; and REP-1 node, to $\tilde{A}_{\psi,3} = \{3, 4, 5, 6, 7\}$.

III. Proposed Nodes

In the following, we identify five nodes and provide their fast decoders. We name them as Type-I, Type-II, Type-III, Type-IV, and Type-V nodes. The proposed decoders can be used for decoding both systematic and non-systematic polar codes designed for any binary-input-symmetric-memoryless channel. For systematic polar codes, the decoder estimates just $\mathbf{x}$, whereas, for non-systematic polar codes, the decoder estimates $\mathbf{u}$ using $\mathbf{u} = \mathbf{x}\mathbf{F}^{\otimes t}$. Also, for the sake of brevity, we will drop the subscripts from $\tilde{A}_{\psi,t}$ and denote the corresponding set of frozen bit-channel indexes by $\tilde{A}^c$. Lastly, we will partition $\mathbf{u}$ and $\mathbf{x}$ in two halves as $\mathbf{u} = \left[\mathbf{u}_0 \ \mathbf{u}_1\right]$, and $\mathbf{x} = \left[\mathbf{x}_0 \ \mathbf{x}_1\right]$, respectively. Using (1), we have $\mathbf{x}_1 = \mathbf{u}_1\mathbf{F}^{\otimes n-1}$, and

$$\mathbf{u}_0\mathbf{F}^{\otimes n-1} = \mathbf{x}_0 + \mathbf{x}_1 = \tilde{\mathbf{x}}. \tag{3}$$

A. Type-I Node

This node corresponds to $\tilde{A} = \{R-2, R-1\}$, $R \geq 4$.

Theorem 1: For Type-I nodes, $\mathbf{x} = x_{\text{R-2}}, x_{\text{R-1}}, \ldots, x_{\text{R-2}}, x_{\text{R-1}}$.

Proof: We prove this result by mathematical induction. First, $\mathbf{x} = x_2, x_3, x_2, x_3$ for $R = 4$ (the base case). For $R > 4$, we presume that the result holds for the block size $R/2$. Since $\tilde{A} = \{R-2, R-1\}$, and $\mathbf{u}_0 = \mathbf{0}$, (3) implies $\mathbf{x}_0 = \mathbf{x}_1$. The result immediately follows by observing that $\mathbf{x}_1 = x_{\text{R-2}}, x_{\text{R-1}}, \ldots, x_{\text{R-2}}, x_{\text{R-1}}$. ∎

Since $x_{\text{R-2}}$ and $x_{\text{R-1}}$ appear on the even-indexed and odd-indexed bits, the optimal decoder for Type-I nodes will make hard decisions on the LLR summations of even-indexed and odd-indexed bits to estimate $x_{\text{R-2}}$ and $x_{\text{R-1}}$, respectively.

It is worth noting that Type-I nodes specialize to 01 and 001 nodes for $R = 4$ and $R = 8$, respectively [8].

B. Type-II Node

This node corresponds to $\tilde{A} = \{R-3, R-2, R-1\}$, $R \geq 4$.

Theorem 2: For Type-II nodes, $x_{\text{R-4}} = x_{\text{R-3}} + x_{\text{R-2}} + x_{\text{R-1}}$, *and* $\mathbf{x} = x_{\text{R-4}}, x_{\text{R-3}}, x_{\text{R-2}}, x_{\text{R-1}}, \ldots, x_{\text{R-4}}, x_{\text{R-3}}, x_{\text{R-2}}, x_{\text{R-1}}$.

Proof: We prove the assertion by mathematical induction. First, $\mathbf{x} = x_1 + x_2 + x_3, x_1, x_2, x_3$ for $R = 4$ (the base case). For $R > 4$, we presume that the result holds for the block size $R/2$. Since $\mathbf{u}_0 = \mathbf{0}$, we have $\mathbf{x}_0 = \mathbf{x}_1$. The result immediately follows by the presumption that $\mathbf{x}_1 = x_{\text{R-4}}, x_{\text{R-3}}, x_{\text{R-2}}, x_{\text{R-1}}, \ldots, x_{\text{R-4}}, x_{\text{R-3}}, x_{\text{R-2}}, x_{\text{R-1}}$. ∎

For Type-II nodes, $x_i = x_j$, where $0 \leq i \leq j \leq R-1$, and $j - i$ is a factor of 4. Consequently, the optimal ML decoder adds the LLRs of the bits with indexes yielding the same remainder when divided by 4. Since $x_{\text{R-4}}, x_{\text{R-3}}, x_{\text{R-2}}, x_{\text{R-1}}$ is a (4, 3) SPC code, these LLRs are then input to an optimal decoder for SPC codes to yield estimates of $x_{\text{R-4}}, x_{\text{R-3}}, x_{\text{R-2}}, x_{\text{R-1}}$. One of the low-complexity optimal decoders for SPC codes is a Wagner decoder [10], which flips the least-reliable bit (the one with the smallest LLR in magnitude) if the parity check is not satisfied by the hard-decision estimates of the bits.

Note that Type-II nodes simplify to the SPC and 0-SPC nodes for $R = 4$ and $R = 8$, respectively [8].

C. Type-III Node

This node corresponds to $\tilde{A}^c = \{0, 1\}$, $R \geq 4$.

Theorem 3: For type-III nodes,

$$(x_0, x_1) = \left(\sum_{i=1}^{R/2-1} x_{2i}, \sum_{i=1}^{R/2-1} x_{2i+1}\right). \tag{4}$$

Proof: Note that when $R = 4$, $\mathbf{x} = x_2, x_3, x_2, x_3$, which satisfies (4). Now, by mathematical induction for $R > 4$, we presume that (4) is satisfied for polar codes of block length $R/2$. Since $\mathbf{u}_0$ is of length $R/2$ and has first two bits frozen to zero, the first two bits of $\tilde{\mathbf{x}}$ (defined in (3)) are

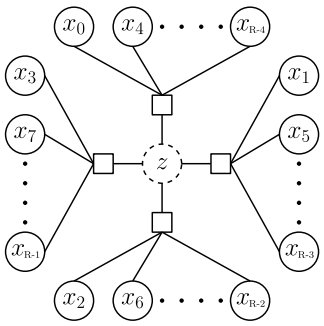

Fig. 2. Tanner graph for a Type-IV node.

given by

$$(\tilde{x}_0, \tilde{x}_1) = \left(\sum_{i=1}^{R/4-1} \tilde{x}_{2i}, \sum_{i=1}^{R/4-1} \tilde{x}_{2i+1} \right). \quad (5)$$

Eq. (4) immediately follows by noting that $\tilde{x}_i = x_i + x_{\frac{R}{2}+i}$. ■

Note that the even-indexed and odd-indexed bits constitute two separate SPC codes. As such, two Wagner decoders can be used to optimally decode the received code vector. Lastly, the Type-III node simplifies to the 01 node when $R = 4$ [8].

D. Type-IV Node

This node corresponds to $\tilde{A}^c = \{0, 1, 2\}$, $R \geq 4$.

Theorem 4: For Type-IV nodes, the first three bits satisfy the following relationship.

$$x_0 + \sum_{i=1}^{R/4-1} x_{4i} = x_1 + \sum_{i=1}^{R/4-1} x_{4i+1} = x_2 + \sum_{i=1}^{R/4-1} x_{4i+2} = z, \quad (6)$$

where $z = \sum_{i=0}^{R/4-1} x_{4i+3}$.

Proof: We use mathematical induction. Clearly, (6) holds when $R = 4$ as $\mathbf{x} = x_3, x_3, x_3, x_3$. For $R > 4$, using (3) and presuming that (6) holds for block size $R/2$, we get

$$\tilde{x}_0 + \sum_{i=1}^{R/8-1} x_{4i} + x_{4i+R/2} = \sum_{i=0}^{R/8-1} x_{4i+3} + x_{4i+3+R/2}, \quad (7)$$

which simplifies to

$$\tilde{x}_0 + x_{R/2} + \sum_{i=1}^{R/4-1} x_{4i} = z. \quad (8)$$

By noting that $\tilde{x}_0 + x_{R/2} = x_0$, we arrive at the first equation in (6). Similarly, other equations can be proved. ■

A non-iterative belief-propagation based optimal MAP decoder [11] for this case can be implemented by noting that an equivalent Tanner graph for this case is cycle free (Fig. 2).

A low-complexity decoder, however, can be implemented by computing $\hat{z}$, a hard-decision estimate of z. Specifically,

$$\hat{z} = \begin{cases} 0, & \text{if } \sum_{i=0}^{3} z_i \geq 0; \\ 1, & \text{otherwise,} \end{cases} \quad (9)$$

where $z_i = 2 \tanh^{-1} \left[\prod_{j=0}^{R/4-1} \tanh \left(\frac{y_{4j+i}}{2} \right) \right]$, and y_i is the LLR of x_i, $i = 0, 1, \ldots, R-1$. After computing $\hat{z}$, we are left with 4 different $(R/4, R/4-1)$ SPC codes. Wagner decoders can then be used to decode the SPC codes.

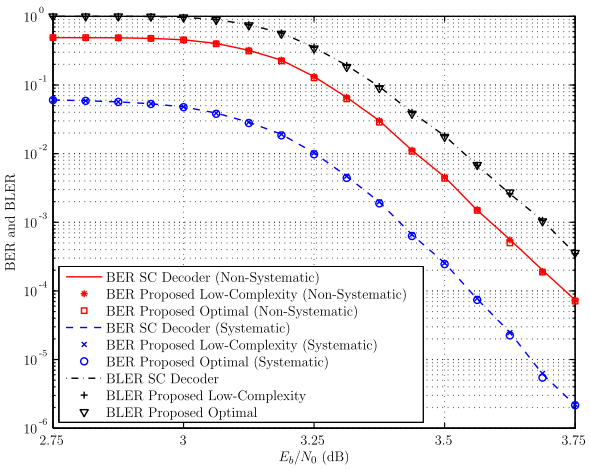

Fig. 3. BER and BLER performance of the designed polar code in AWGN channels when decoding is performed by the conventional SC and the proposed decoders.

Note that Type-IV nodes simplify to REP and REP-1 nodes for $R = 4$ and $R = 8$, respectively [8].

E. Type-V Node

This node corresponds to $\tilde{A} = \{R-5, R-3, R-2, R-1\}$ for $R \geq 8$.

Theorem 5: For Type-V nodes, $x_{R-8} = x_{R-5} + x_{R-3} + x_{R-2}$, $x_{R-7} = x_{R-5} + x_{R-3} + x_{R-1}$, $x_{R-6} = x_{R-5} + x_{R-2} + x_{R-1}$, $x_{R-4} = x_{R-3} + x_{R-2} + x_{R-1}$, *and* $x_i = x_j$ *for* $0 \leq i \leq j \leq R-1$, *where* $j - i$ *is a multiple of 8.*

Proof: We employ mathematical induction to prove this. For $R = 8$, $\mathbf{x} = x_3 + x_5 + x_6, x_3 + x_5 + x_7, x_3 + x_6 + x_7, x_3, x_5 + x_6 + x_7, x_5, x_6, x_7$, which corresponds to a Type-V node. Since $\mathbf{u}_0 = \mathbf{0}$ for $R > 8$, we have $\mathbf{x}_0 = \mathbf{x}_1$. Therefore, by presuming that $\mathbf{x}_1$ corresponds to a Type-V node, $\mathbf{x} = [\mathbf{x}_1 \ \mathbf{x}_1]$ also corresponds to a Type-V node. ■

An optimal-ML decoder for Type-V nodes can easily be implemented [12]. Noting that the Type-V node specializes to the REP-SPC node for $R = 8$, we can use the low-complexity decoder proposed in [6] for Type-V nodes. Specifically, LLRs of the bits whose indexes differ by a multiple of 8 are added. Denoting the ith element of the added LLRs, $\mathbf{y}$, by y_i, $i = 0, 1, \ldots, 7$, the decoder computes $\hat{z}$ as

$$\hat{z} = \begin{cases} 0, & \text{if } \sum_{i=0}^{3} y_i \circledast y_{i+4} \geq 0; \\ 1, & \text{otherwise.} \end{cases} \quad (10)$$

Afterwards, $(1-2z)y_0 + y_4, \ldots, (1-2z)y_3 + y_7$ is input to a Wagner decoder to estimate $x_4, \ldots, x_7$. Lastly, x_i is estimated by $\hat{x}_i = \hat{x}_{i+4} + \hat{z}$, for $i = 0, 1, 2, 3$.

IV. Results

We compare the decoding performance and latency of the proposed decoders with the SC and fast-SSC decoders below.

A. Decoding Performance

Fig. 3 depicts the bit-error rate (BER) and block-error rate (BLER) of a $(32768, 27648)$ polar code designed for an additive white Gaussian noise (AWGN) channel

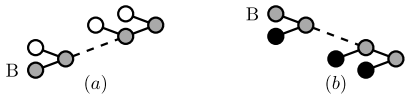

Fig. 4. Decoding trees of the proposed nodes correspond to either decoding tree (a) or decoding tree (b).

by Tal & Vardy algorithm [13] for a design $E_b/N_0 = 3.75$ dB. For comparison, both the optimal and sub-optimal fast decoders are implemented for the proposed nodes. Observe that all decoders show similar performance.

B. Decoding Latency

We compare the decoding latency of the proposed and fast-SSC decoders with the following presumptions. First, hard decision on LLRs and bit operations are carried out instantaneously. Second, addition/subtraction of real numbers, check-node operation and Wagner decoding consume one clock cycle. Note that similar assumptions were made in [5] to assess the latency of rate-1 node decoders.

The decoding trees of Type-I, Type-II, and Type-V nodes are of the form given in Fig. 4a, whereas the decoding trees of Type-III and Type-IV nodes correspond to Fig. 4b. The rate-0 and rate-1 nodes are white and black, respectively. The node B is 001 for Type-I, 0-SPC for Type-II, 01 for Type-III, REP-1 for Type-IV, and REP-SPC for Type-V nodes.

Observe that computing LLRs for either child node consumes 1 cycle. Further, if a child is a rate-0 or rate-1 node, the decoding can be performed instantaneously [5]. For Type-I, Type-II and Type-V nodes, the left children are rate-0 nodes. As such, LLRs of only right children is computed for the fast-SSC decoding. Consequently, $t-3$ cycles are needed to reach node B, which can be decoded in 1 cycle [6], [8]. Therefore, the decoding latency for these nodes under fast-SSC decoding is $t-2$ cycles.

For Type-III and Type-IV nodes, the LLRs for left-children are also computed. Further, after decoding the left child, one cycle is consumed for computing LLRs for the right child. Also, node B can be decoded in one cycle [6], [8]. Since the size of node B is 4 and 8 for Type-III and Type-IV nodes, respectively, fast-SSC decoding of these nodes will incur delays of $2t-3$ and $2t-5$ cycles, respectively.

Since the proposed decoders do not compute LLRs for the left and right children, their decoding latency is reduced. For example, for Type-I node, the latency is just 1 cycle as the LLRs can be added in one cycle. For Type-II nodes, the decoding delay is 1 cycle when $R \leq 8$ [6], [8]. For $R > 8$, the delay is 2 cycles as the LLR summation takes 1 cycle, and Wagner decoder consumes another cycle. Furthermore, due to Wagner decoding, the decoding latency of the proposed decoders for Type-III nodes is only 1 cycle. For Type-IV nodes, z can be estimated in 2 cycles (1 for performing check-node operation and 1 for adding the LLRs). Also, Wagner decoding can be performed in parallel with estimating z presuming $z = 0$ or 1. As such, the decoding latency is 2 cycles. Lastly, since a REP-SPC node can be decoded in 1 cycle [6], [8], the decoding latency of Type-V is 1 cycle when $R = 8$ and 2 cycles when $R > 8$.

TABLE I

DECODING LATENCY OF THE PROPOSED NODES FOR $R = 2^t$

Node	**Type-I**	**Type-II**	**Type-III**	**Type-IV**	**Type-V**
Fast-SSC	$t-2$	$t-2$	$2t-3$	$2t-5$	$t-2$
Proposed	1	1 or 2	1	2	1 or 2

TABLE II

THE COUNT AND $R_{\max}$ OF THE PROPOSED NODES FOR A POLAR CODE DESIGNED FOR THE BEC WITH ERASURE PROBABILITY 0.3

N	**Rate**	**Type-I**	**Type-II**	**Type-III**	**Type-IV**	**Type-V**
4096	5/8	5, 32	8, 128	2, 8	6, 128	22, 64
8192	43/64	7, 128	13, 128	8, 64	12, 128	27, 256
16384	9/16	8, 32	11, 16	11, 128	11, 256	119, 128

Table I compares the decoding latency of the proposed with that of the fast-SSC decoders for the proposed nodes.

Table II shows the number of occurrences and the maximum size, $R_{\max}$, of the proposed nodes (in the form of count, $R_{\max}$) for different-length polar codes designed for a binary-erasure channel (BEC). Table I shows that large values of R result in high gains in the decoding speed. As such, our proposed decoders can significantly improve the decoding latency.

V. CONCLUSION

In this work, we identified five nodes in the code tree of a polar code and presented their fast decoders. The proposed decoders can significantly improve the decoding speed of a polar code without affecting the bit-error-rate and block-error-rate performance.

REFERENCES

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

[2] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2011, pp. 1–5.

[3] *Performance Study of Polar Code Candidates*, document 3GPP TSG-RAN WG1 #88, R1-1703538, Ericsson, Athens, Greece, Feb. 2017.

[4] K. Niu, K. Chen, J. Lin, and Q. T. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 192–203, Jul. 2014.

[5] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.

[6] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, May 2014.

[7] G. Sarkis, I. Tal, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Flexible and low-complexity encoding and decoding of systematic polar codes," *IEEE Trans. Commun.*, vol. 64, no. 7, pp. 2732–2745, Jul. 2016.

[8] P. Giard, A. Balatsoukas-Stimming, G. Sarkis, C. Thibeault, and W. J. Gross, "Fast low-complexity decoders for low-rate polar codes," *J. Signal Process. Syst.*, to be published, doi: 10.1007/s11265-016-1173-y.

[9] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, "237 Gbit/s unrolled hardware polar decoder," *Electron. Lett.*, vol. 51, no. 10, pp. 762–763, May 2015.

[10] R. Silverman and M. Balser, "Coding for constant-data-rate systems," *Trans. IRE Prof. Group Inf. Theory*, vol. 4, no. 4, pp. 50–63, Sep. 1954.

[11] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[12] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 963–975, Sep. 1989.

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