Journal of Semiconductor Engineering Vol. 1, Issue 2, September 2020
https://doi.org/10.22895/jse.2020.0005
Received 1 Jun 2020, Revised 19 Jun 2020, Accepted 24 Jun 2020 ISSN(On-line) 2733-6719

Recent Successive Cancellation Decoding Methods for Polar Codes

Soyeon Choi[1], Youngjoo Lee[2], and Hoyoung Yoo[1]

[1] Electronics Engineering, Chungnam National University, Daejeon, 34134, Korea
[2] Electrical Engineering, Pohang University of Science and Technology, 37673, Korea

Corresponding Author: Hoyoung Yoo (hyyoo@cnu.ac.kr)

Funding Information: This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2019M3F3A1A01074449)

ABSTRACT

Due to its superior error correcting performance with affordable hardware complexity, the Polar code becomes one of the most important error correction codes (ECCs) and now intensively examined to check its applicability in various fields. However, Successive Cancellation (SC) decoding that brings the advanced Successive Cancellation List (SCL) decoding suffers from the long latency due to the nature of serial processing limiting the practical implementation. To mitigate this problem, many decoding architectures, mainly divided into pruning and parallel decoding, are presented in previous manuscripts. In this paper, we compare the recent SC decoding architectures and analyze them using a tree structure.

KEY WORDS

Polar codes, successive cancellation decoding, error correction codes, low-latency processing, 5G communications.

1. INTRODUCTION

Polar code is a channel coding technique introduced by Arikan in 2008. It is the first error correction code that can provably achieves Shannon's channel capacity in a memoryless channel [1]. Because of superior ability to achieve Shannon's channel capacity, the polar code has become one of the most important error correction codes in a code theory field and can be applied to error correction in next-generation communication and data storage systems [2], [3].

The conventional decoding for polar codes is successive cancellation (SC) decoding [1] based on divide-and-conquer approach that interchanges soft and hard information iteratively. Since SC decoding employs the previously estimated bits, SC decoding should decode a bit in a successive manner [1]. Based on the conventional SC decoding [1], advanced decoding schemes including successive-cancellation-list (SC-List) decoding [4], [5], and successive-cancellation-flip (SC-Flip) decoding [6], [7] are introduced to improve the error correcting performance even for short and moderate polar codes. In particular, SCL decoding has been adopted in 3GPP 5G wireless communication protocol [8]. However, SC decoding [1] inevitably suffers from long decoding latency due to the nature of sequential processing, which limits the practical implementation.

To solve this problem, a number of studies have investigated in previous works, and they are mainly divided into pruning and parallel approaches [9]–[15]. In general, pruning approach tries to eliminate unnecessary nodes and replace a complex computation with a simple computation. It takes the most important role for pruning approach to determine which nodes are possible to be pruned. For the first time, [9] introduces the pruning-based SC decoding by eliminating all-frozen-nodes and all-information-nodes. [10] presents a method to prune mixed nodes further using maximum likelihood (ML) detection. Since ML detection demands severe computations, [10] has a limitation in that it is impractical. Whereas [10] tries to eliminate all the mixed node, [11] selects and focuses on particular nodes with specific patterns. [11] has a point in that it is

succeeded in a practical implementation. Moreover, [12] presents a channel adaptive pruning method by using the widely known syndrome check technique. Unlike the previous pruning-based algorithms with a fixed pruning capability, [12] provides a flexible pruning capability depending on the channel condition. Although the pruning approach can reduce decoding +latency by eliminating unnecessary computations, its nature of sequential processing is still inevitable. To mitigate this limitation, parallel approach is introduced in [13] and [14]. [13] computes two bits at a time by considering all the combination of two bits simultaneously. Furthermore, [14] introduces a method to parallelize p-bit by decomposing the original decoding [1]. First, p soft information is computed individually, and next they are merged by selecting the most likely one among 2^p candidates.

In this paper, we aim to provide informative intuition on the recent SC decoding used for polar decoder design. The rest of this paper focuses on comparing and analyzing the previous decoding based on a tree structure. Section 2 describes backgrounds including basic encoding and SC decoding. Section 3 and 4 reviews pruning-based SC decoding [9]–[12] and parallel SC decoding [13]–[15] respectively. Section 5 discusses advantages and limitations of the recent SC decoding and concludes this paper.

2. BACKGROUNDS

A. POLAR CODE ENCODING

A polar code (N, K) is a type of linear block code that transmits N codeword bits given K information bits. Based on channel polarization phenomenon [1], the message vector $\mathbf{u} = [u_0, u_1, \ldots, u_{N-1}]$ is firstly generated by assigning K information bits to K most reliable channels over N channels. The remaining (N–K) bits, i.e. frozen bits, are set to a fixed value known to both transmitter and receiver. After assigning K information bits and (N–K) frozen bits, a $1\times N$ codeword $\mathbf{x}$ is computed by multiplying the $1\times N$ message vector $\mathbf{u}$ and the $N\times N$ generated matrix $\mathbf{G^n}$. i.e. $\mathbf{x} = \mathbf{uG^n}$. Note that $n=\log_2 N$ and $\mathbf{G^n}$ is the n-th Kronecker power to the kernel matrix $\mathbf{G}^1 = [1\ 0; 1\ 1]$. For instance, when n=4 and N=16, the codeword $\mathbf{x}$ can be obtained as (1).

$$\mathbf{x} = \mathbf{u}\cdot\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&0&1&1&0&0&0&0&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}, \quad (1)$$

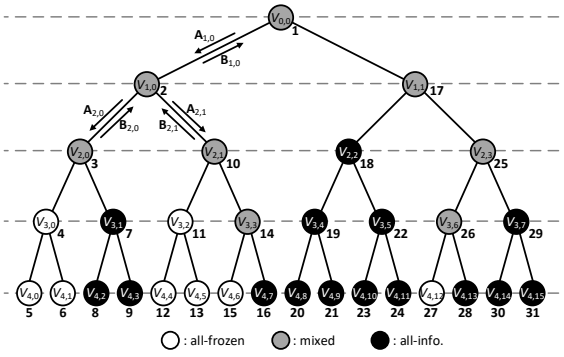


Figure 1. Successive-cancellation decoding tree for (16, 10) polar codes

After the codeword generation, a transmitter sends a $1\times N$ codeword $\boldsymbol{x}$ through a noisy channel $\mathbf{n}$, and then a receiver receives a $1\times N$ noisy codeword $\boldsymbol{y} = \boldsymbol{x}+\mathbf{n}$.

B. SUCCESSIVE CANCELLATION DECODING

The conventional decoding method is SC decoding [1] that estimates a message bit from 0 to N–1 in a sequential manner. To clarify SC decoding [1] for Polar (N, K) codes, it is general to adopt a binary decoding tree with a $n = \log_2 N$ depth. For instance, Figure. 1 depicts 4-depth binary tree for Polar (16, 10) codes, where the white, black, and grey nodes represent nodes having all-information, all-frozen, and mixed leaves, respectively.

Let us assume that node $V_{i,j}$ be the j-th node at the i-th stage on the binary decoding tree. When the root node is indexed from the stage 0, the root node becomes $V_{0,0}$ and contain 16 leaf nodes. It is noticeable that $V_{i,j}$ at the stage j includes $L_i = 2^{n-i}$ leaf nodes. For instance, $V_{1,0}$ and $V_{1,1}$ include $8 = 2^{4-1}$ leaf nodes.

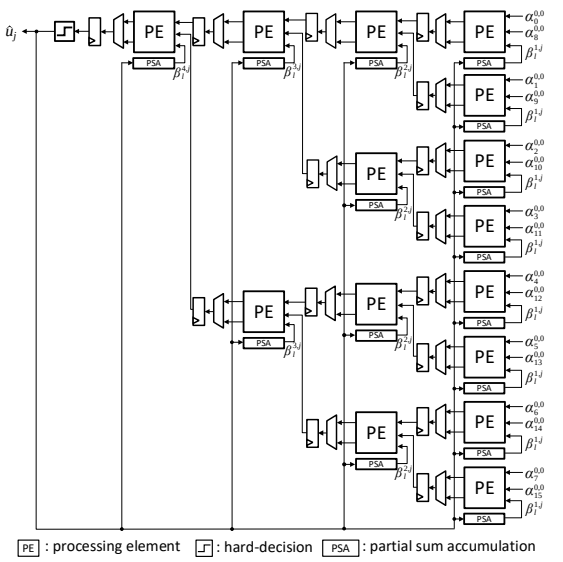
For decoding process, each node interchange soft and hard information successively through the graphs on the binary tree. A node $V_{i,j}$ receives a soft information vector $\mathbf{A}_{i,j} = [\alpha_0^{i,j}, \alpha_1^{i,j}, \ldots, \alpha_{L_i-1}^{i,j}]$ from its parent node $V_{(i-1),\lceil j/2\rceil}$ and returns a hard information vector $\mathbf{B}_{i,j}=[\beta_0^{i,j}, \beta_1^{i,j}, \ldots, \beta_{L_i-1}^{i,j}]$. Given the $\mathbf{A}_{i,j}$ from the parent node $V_{(i-1),\lceil j/2\rceil}$, a node $V_{i,j}$ computes soft information vectors $\mathbf{A}_{i+1,2j}$ and $\mathbf{A}_{i+1,2j+1}$ as follows.

$$\begin{aligned}\alpha_l^{i+1,2j} &= F(\alpha_l^{i,j}, \alpha_{l+L_i+1}^{i,j})\\ \alpha_l^{i+1,2j} &= G(\alpha_l^{i,j}, \alpha_{l+L_i+1}^{i,j}, \beta_l^{i+1,2j})\end{aligned}, \quad (2)$$

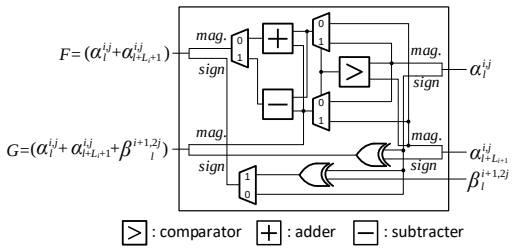
where l is a vector index ranging from 0 to L_{i-1}. F and G functions are generally defined as (3) to prohibit a complex computation.

$$\begin{aligned}F(x,y) &= \mathrm{sgn}(x)\mathrm{sgn}(y)\min(|x|,|y|)\\ G(x,y,u) &= (-1)^u x + y\end{aligned}, \quad (3)$$

where sgn(x) is a sign function that results in 1 for $x>0$ and -1 for otherwise. Next, the computed soft information vector $\mathbf{A}_{i+1,2j}$ and $\mathbf{A}_{i+1,2j+1}$ are transferred to the left child node $V_{i+1,2j}$ and right child $V_{i+1,2j+1}$, respectively. When information vector $\mathbf{B}_{i+1,2j}$ and $\mathbf{B}_{i+1,2j+1}$ are received from the left child node $V_{i+1,2j}$ and right child $V_{i+1,2j+1}$, $V_{i,j}$ newly computes hard information vector $\mathbf{B}_{i,j}$ as (4) and returns $\mathbf{B}_{i,j}$ to the parent node $V_{(i-1),\lceil j/2 \rceil}$.



(a)



(b)

Figure 2. (a) The tree hardware architecture for SC decoding for polar code when N = 16 and (b) PE architecture

$$\begin{aligned} \beta_l^{i,j} &= \beta_l^{i+1,2j} \oplus \beta_l^{i+1,2j+1} \\ \beta_{l+L_i+1}^{i,j} &= \beta_l^{i+1,2j+1} \end{aligned} . \tag{4}$$

To initiate SC decoding [1], the soft information $\mathbf{A}_{0,0}$ at the root node $V_{0,0}$ should be initialized using log-likelihood ratio as

$$\alpha_l^{0,0} = \log\left(\frac{\Pr(y_l \mid x_l = 0)}{\Pr(y_l \mid x_l = 1)}\right), \; 0 \le l \le N-1. \tag{5}$$

In addition, the leaf nodes $V_{n,j}$ for $0 \le j \le N-1$ should determines hard-information-vector $\mathbf{B}_{n,j}$ to operate SC decoding [1] successively. The hard information vector $\mathbf{B}_{n,i}$ is computed as

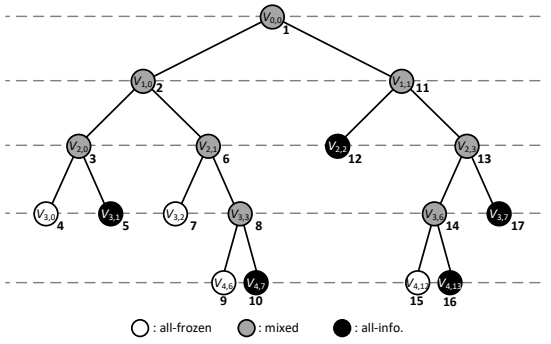


Figure 3. Simplified SC decoding tree [9] for Polar (16, 10)

$$\hat{u}_j = \beta_0^{n,j} = \begin{cases} 0, \text{ if } j \in Fr \text{ or } \alpha_0^{n,j} \ge 0 \\ 1, \text{ otherwise} \end{cases}, \tag{6}$$

where Fr indicates the set of indices for frozen bits. When all the leaf node estimates a message bit, the conventional SC decoder [1] finishes, and consequently returns the decoded output $\hat{\mathbf{u}} = [\beta_0^{n,0}, \beta_1^{n,1}, ..., \beta_{L_i-1}^{n,N-1}]$. Furthermore, the typical tree-based structure [1] of SC decoding is depicted in Figure 2(a). Due to the binary tree structure, it demands $N-1$ processing elements (PEs), $N-1$ partial sum accumulators (PSAs), and one hard-decision module for Polar (N, K) codes. Figure 2(b) shows the internal structure of PE in more details. Given soft information vector $\mathbf{A}_{i,j}$ and hard information vector $\mathbf{B}_{i,j}$, the PE computes soft information vectors $\mathbf{A}_{i+1,2j}$ and $\mathbf{A}_{i+1,2j+1}$ according to F and G function of (3), respectively. PSA is responsible for computing hard information vectors $\mathbf{B}_{i,}$ as (4) and hard decision module computes the estimated output $\hat{\mathbf{u}}$ as (6).

3. RECENT PRUNING-BASED SC DECODING

Due to the use of partial sum $\mathbf{B}_{i,j}$ to compute function G in the conventional SC decoding [1], a node $V_{i,j}$ needs to wait until all the previous bits are estimated. For instance, the numbers next to the node in Figure 1 indicate the visiting order, which is directly proportional to the decoding latency. When a SC decoder [1] can operation one node at one clock cycle, 31 clock cycles are necessary to decoder Polar (16, 10) using the binary decoding tree in Figure 1. In general, a typical SC decoder [1] for Polar (N, K) codes demands ($2N-1$) clock cycles, and thus as the code length N increases, the conventional SC decoder more suffers

from a long decoding latency due to a serialized

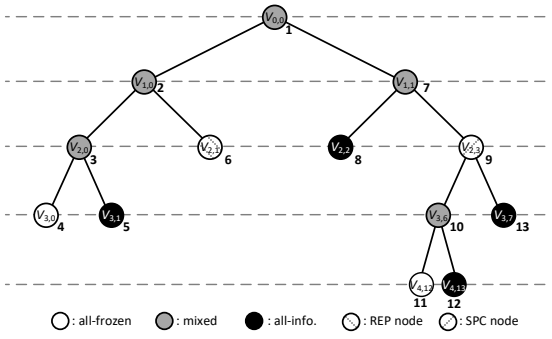


Figure 4. Fast-SSC decoding tree [11] for Polar (16, 10) codes

operation.

To reduce the decoding latency, [9] presents simplified SC (SSC) decoding that eliminates node computations by pruning the all-information-nodes and all-frozen-nodes. More precisely, it is trivial that all-frozen-nodes can be pruned since all the frozen leaf nodes associated with a any parent all-frozen-node can be directly estimated as zero without visiting intermediate child nodes successively. Furthermore, SSC [9] proves that all-information-nodes in addition to all-frozen-nodes can be pruned by replacing an iterative and complex computations with a simple hard decision function. Figure 3 shows the example of SSC [9] for Polar (16, 10) codes. All-frozen-node $V_{3,0}$ and $V_{3,2}$ and all-information-node $V_{2,2}$, $V_{3,1}$ and $V_{3,7}$ are pruned. The decoding latency in the SSC decoders [9] becomes 15 from 31 by efficiently pruning unnecessary nodes with simple computation replacement. To sum up, SSC [9] firstly presents a pruning-based SC decoding that skips all-information-nodes and all-frozen-nodes completely. ML-SSC [10] newly includes the mixed-nodes as pruning candidate nodes. Whereas SSC [9] can prune only all-information-nodes and all-frozen-nodes, ML-SSC [10] presents a method to prune the mixed-nodes further. ML-SSC [10] performs maximum likelihood (ML) detection that computes each likelihood for all possible vectors and determines the most likely one as an estimated vector. Although ML-SSC [10] enlarges the pruning nodes including the mixed-nodes, ML-SSC [10] cannot be directly implemented because of severe hardware complexity due to sorting and arithmetic operations in ML detection. As a pruning candidate node at an upper stage, hardware complexity for pruning a mixed-node increases exponentially.

ML-SSC [10] tries to present an efficient method to prune all the mixed-node but it is impractical due to severe hardware complexity. Fast-SSC [11] proposes a pruning method that eliminate nodes not all the mixed-node but some selected nodes with specific frozen patterns. Fast SSC [11] examines hardware complexity required to compute ML decision for all frozen patterns and selects some patterns that relax the computing overhead of ML decision. Fast-SSC defines two frozen patterns; repetition (REP) and single-parity-check (SPC) [11]. REP represents a frozen-pattern that only the last leaf node is an information-node and the others are frozen-nodes, and SPC represents a frozen pattern that only the first leaf node is a frozen-node and the others are information-nodes. As a result, Fast-SSC [11] prunes several mixed-nodes associated with REP and SPC frozen patterns by replacing original complex computations of F and G in (3) with a simple computation such as XOR. It is noticeable that REP and SPC patterns are dominant in typical Polar codes, so that Fast-SSC [11] dramatically reduces hardware computations with a small hardware increase used for REP and SPC computation. Figure 4 shows the

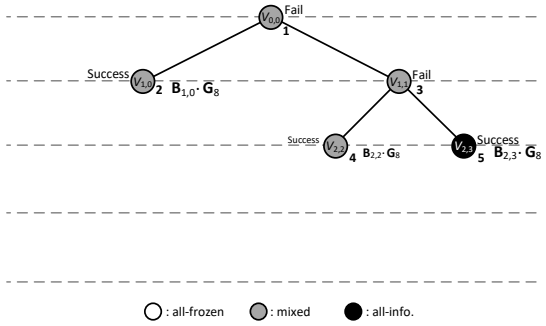


Figure 5. Syndrome-Check SC decoding tree [12] for Polar (16, 10) codes

example of Fast SSC [11] for Polar (16, 10) codes. All-frozen-node $V_{3,0}$ and all-information-node $V_{3,1}$ and $V_{3,7}$ are pruned and node $V_{2,1}$ and $V_{2,3}$ are replaced with REP node and SPC node, respectively.

Pruning ratios in all the previous pruning-based decoding [9]–[11] are determined when the target polar codes are constructed based on the channel polarization phenomenon [1]. However, channel condition affects the initial condition on the received vector **y**, it is important to support an adaptive pruning scheme for the channel condition. [12] presents syndrome-check-SC (SC-SC) [12] decoding that skips unnecessary sub-trees if the syndrome check of a constituent code is satisfied. For the syndrome check, SC-SC [12] employs the generator matrix associated with the corresponding subtree, and the recursive operation associated with such a constituent code is replaced with simple computations without sacrificing the error-rate performance. Figure 5 depicts the example of SC-SC decoding [12] for Polar (16, 10) codes. In the example, $V_{0,0}$ and $V_{1,1}$ fail the syndrome check leading to the continuous child nodes visiting, but $V_{1,0}$, $V_{2,2}$, and $V_{2,3}$ succeed in the syndrome check and result in subtree pruning. When a channel SNR is high, SC-SC [12] can prune more of the decoding tree since the probability to

Table 1. Summary of the recent SC decoding comparison

Algorithm	SSC [9]	ML-SSC [10]	Fast-SSC [11]	SC-SC [12]	2b-SC [13]	*p*b-SC [14]
Type	Pruning	Pruning	Pruning	Pruning	Parallel	Parallel
Pruning Ratio	Small	High	High	Moderate	N.A.	N.A.
Parallel Factor	N.A.	N.A.	N.A.	N.A.	2	Any Factor
Additional hardware	Small	High	Moderate	Moderate	Moderate	High
Channel Adjustability	No	No	No	Yes	No	No

satisfy the syndrome check becomes higher, and when a channel SNR is low, SC-SC [12] can prune less of the decoding tree. As a result, SC-SC [12] can provide a channel adaptive pruning method compared to the previous fixed pruning methods.

4. RECENT PARALLEL SC DECODING

The previous pruning methods succeed in reducing a decoding latency in SC decoders [9]–[12], however, they inevitably demand successive orders since the basic operation is performed based on the original SC decoding tree [1]. To alleviate the limitation of the sequential processing in the previous pruning-based SC decoding [9]–[12], the parallel SC decoding approach is also proposed in [13] and [14].

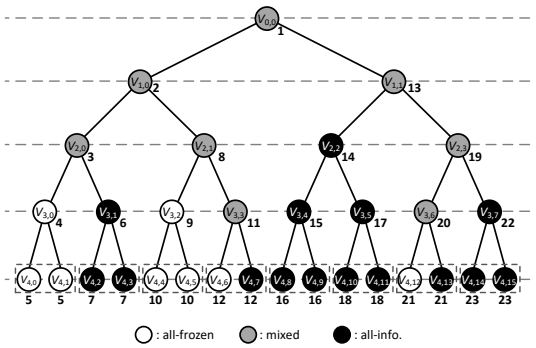


Figure 6. 2-bit parallel SC decoding tree [13] for Polar (16, 10) codes

For the first time, 2b-SC decoding in [13] modifies the serial processing in the conventional SC decoding [1] into the 2-bit parallel processing by combining two leaf nodes having the same parent. For two leaf nodes, the combination can be (F, F), (F, I), (I, F), and (I, I), where F and I represent frozen and information leaf nodes respectively. More precisely, the leaf nodes for (F, F) directly estimate as zero, and the leaf nodes for (I, I) are simply computed using a simple hard decision operation, which is similar to SSC [9]. In addition, the leaf nodes for (I, F) and (F, I) are treated similarly as SPC and REP in Fast-SSC [11]. For instance, Figure 6 shows the example of 2b-SC [13] for Polar (16, 10) codes. As shown in Figure 6, two leaf nodes with the same parent can be operated at the same time leading to the latency reduction from 31 of the original decoding latency to 23. As a result, 2b-SC [13] can decrease a decoding latency by computing two leaf nodes simultaneously.

Furthermore, *p*b-SC [14] introduces a method to parallelize *p* small-sized subtrees by decomposing the original SC decoding tree. First, *p* small-sized subtrees compute soft information individually, and after generating *p* soft information the merging node computes the most likely *p*-bit pattern for possible 2^p candidates. For instance, 4-bit parallel SC decoding tree of *p*b-SC [14] for Polar (16, 10) codes are exemplified in Figure 7. As shown in Figure 7, 4-subtrees with the depth of 2 are operated simultaneously and the merging node computes an estimated pattern given 4 soft

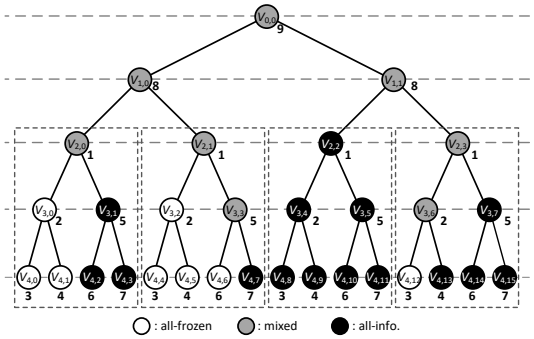


Figure 7. 4-bit parallel SC decoding tree [14] for Polar (16, 10) codes

information, each of which is generated by a leaf node from a different sub tree. It is important to notice that the merging function in *p*b-SC [14] is basically similar to ML detection that computes all possible 2^p candidates. As ML-SSC [10] is impractical due to the severe hardware complexity, *p*b-SC [14] is not practical especially when the parallel factor *p* is not small. It is hard to increase the parallel factor more than 4, and this limited parallel factor is not desirable for most polar codes with a long codeword. Hardware-friendly-*p*b-SC [15] relaxes this problem in a similar approach to Fast-SSC [11]. As Fast-SSC [11] selects some particular frozen patterns that can relax ML detection in ML-SSC

[10], hardware-friendly-*p*b-SC [15] also selects some particular frozen patterns that can relax ML detection in *p*b-SC [14]. As a result, hardware-friendly-*p*b-SC [15] presents a practical *p*-parallel structure by minimizing the overheads for estimating decoded bits in merging function.

5. DISCUSSION & CONCLUSION

Table 1 summarizes the recent SC decoding for Polar codes, which are mainly divided into two main approaches, pruning-based and parallel approaches. For pruning-based approach, SSC [9] first attempts to remove unnecessary nodes such as all-information-nodes and all-frozen-nodes, but it cannot eliminate mixed-nodes at all. ML-SSC [10] enlarges unnecessary nodes further by including mixed-nodes, and it provides a method to replace an original complex computation with ML function. However, ML-SSC [10] still considers as an impractical method due to severe hardware complexity resulting from ML detection. To mitigate this problem and provide a practical decoding algorithm, Fast-SSC [11] selects particular mixed-nodes among all possible mixed-nodes as removable nodes by taking the required computations for ML detection into account. In addition, SC-SC [12] introduces a new pruning method that can adjust a channel condition by checking the syndrome. On the other hand, for parallel approach, 2b-SC [13] and *p*b-SC [14] improves the previous pruning approach, which has a limitation resulting from the nature of sequential processing. 2b-SC [13] modifies the leaf nodes computation to generate two bits at the same time by considering all the cases simultaneously. *p*b-SC [14] decomposes the original decoding tree into *p* small-sized sub trees. After *p* soft information is generated individually, they are merged to construct original soft information. Since this paper aims to provide intuitive explanation about SC decoding for polar codes, we tried to depict graphics rather than to use mathematics. For more details, it is recommended to investigate the reference. Further research on SC decoding seems still necessary, not only for improving latency but also for increasing throughput.

ACKNOWLEDGMENT

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2019M3F3A1A01074449)

REFERENCES

[1] E. Arikan. "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051-3073, Jul. 2009.

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

[3] 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, Jan. 2013.

[4] I. Tal and A. Vardy, "List Decoding of Polar Codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213-2226, May 2015.

[5] B. Li, H. Shen and D. Tse, "An Adaptive Successive Cancellation List Decoder for Polar Codes with Cyclic Redundancy Check," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2044-2047, Dec. 2012.

[6] O. Afisiadis, A. Balatsoukas-Stimming and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," *2014 48th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2014, pp. 2116-2120.

[7] L. Chandesris, V. Savin and D. Declercq, "An Improved SCFlip Decoder for Polar Codes," *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016, pp. 1-6.

[8] 3GPP, ″5G: Study on new radio (NR) access technology," 3GPP TS 38.212 v.15.0.0, 2017.

[9] A. Alamdar-Yazdi and F. R. Kschischang, "A Simplified Successive-Cancellation Decoder for Polar Codes," *IEEE Communications Letters*, vol. 15, no. 12, pp. 1378-1380, Dec. 2011.

[10] G. Sarkis and W. J. Gross, "Increasing the Throughput of Polar Decoders," *IEEE Communications Letters*, vol. 17, no. 4, pp. 725-728, Apr. 2013.

[11] G. Sarkis, P. Giard, A. Vardy, C. Thibeault and W. J. Gross, "Fast Polar Decoders: Algorithm and Implementation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 946-957, May 2014.

[12] H. Yoo and I. Park, "Efficient Pruning for Successive-Cancellation Decoding of Polar Codes," *IEEE Communications Letters*, vol. 20, no. 12, pp. 2362-2365, Dec. 2016.

[13] B. Yuan and K. K. Parhi, "Low-Latency Successive-Cancellation Polar Decoder Architectures Using 2-Bit Decoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1241-1254, Apr. 2014.

[14] H. S. B. Li and D. Tse. (Sep, 2013) "Parallel decoders of polar codes," Sep. 2013 [Online]. Available: http://arxiv.org/abs/1309.1026.

[15] D. Kam and Y. Lee, "Ultra-Low-Latency Parallel SC Polar Decoding Architecture for 5G Wireless Communications," *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, 2019, pp. 1-5

AUTHOR BIOGRAPHIES

Soyeon Choi received the B.S. degree in electronics engineering from Chungnam National University, Daejeon, Korea, in 2018. She is working toward the Unified Master and Ph.D. degree in Electronics Engineering, Chungnam National University, Daejeon, Korea.

Her main interests are VLSI for error correction codes and FPGA reconfiguration.

Youngjoo Lee received the B.S., M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2008, 2010 and 2014, respectively. Since February 2017, he has been an Assistant Professor in the department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea. Prior to joining POSTECH, he was with Interuniversity Microelectronics Center (IMEC), Leuven, Belgium, from May 2014 to February 2015, where he researched reconfigurable SoC platforms for software-defined radio systems. From March 2015 to February 2017, he was with the Faculty of the Department of Electronic Engineering, Kwangwoon University, Seoul, Korea.
His current research interests include the algorithms and architectures for embedded processors, intelligent multimedia systems, advanced error-correction codes, and next-generation wireless systems.

Hoyoung Yoo received the B.S. degree in electrical & electronics engineering from Yonsei University, Seoul, Korea, in 2010. He received the M.S. and Ph.D. degree in electronic engineering from KAIST in 2012 and 2016. Since 2016, he has been with the department of Electronics Engineering, Chungnam National University (CNU), Daejeon, Korea, where he is now an Assistant Professor. Prior to joining CNU, he was with Samsung Electronics, Hwasung, Korea from March 2016 to August 2016, where he researched non-binary LDPC decoder for NAND flash memories.
His research interests are algorithms and architectures for error correction codes, FPGA reverse engineering, GNSS communication, and 5G communication systems.