I. INTRODUCTION

A. Motivations and Background

Higher throughput has always been a primary target along the course of mobile communications evolution. Driven by high data rate applications such as virtual/augmented reality (VR/AR) applications, the sixth generation wireless technology (6G) requires a peak throughput of 17Tbps [2]. This is roughly a $50 \times \sim 100 \times$ increase over the $10 \sim 20\text{Gbps}$ target throughput for 5G standards.

To support such a high data rate, we need new physical layer design to further reduce implementation complexity, save energy, and improve spectral efficiency. This is particularly true when the peak throughput requirement is imposed on a resource constrained (limited processing power, storage, and energy supply etc.) device. Since channel coding is well-known to consume a substantial proportion of baseband computational resources, it poses a bottleneck for extreme throughput. To this end, channel coding is a key physical layer technologies in order to guarantee 17Tbps peak throughput for 6G.

Polar codes, defined by Arıkan in [3], are proved to achieve capacity on discrete binary memoryless symmetric (BMS) channels under successive cancellation (SC) decoding. At finite length, SC decoding provide moderate performance with a very low complexity [4].

Although the SC decoding algorithm seems unsuitable for high-throughput applications due to its serial nature, state-of-the-art SC decoders [5] [6] [7] [8] [9] managed to significantly simplify and parallelize the decoding process such that the area efficiency of SC decoding has exceeded that of belief propagation (BP) decoding for low-density parity-check codes (LDPC). In particular, these works represent SC decoding as a binary tree traversal [6], as shown in Fig. 1(a). Each subtree therein represents a shorter polar code. The original SC decoding algorithm traverses the tree by visiting all the nodes and edges, leading to high decoding latency. Simplified SC decoders can fast decode certain subtrees (shorter polar codes) and thus “prune” those subtrees. The resulting decoding latency is largely determined by the number of remaining edges and nodes in the pruned binary tree. Several tree-pruning techniques have been proposed in [6], [10] and [11]. To achieve 17Tbps throughput, more aggressive techniques need to be proposed on both the decoding and encoding sides.

B. Contributions

This paper introduces a novel polar code construction method, coined as “fast polar codes”, to facilitateparallelized processing at an SC decoder. In contrast to some existing decoding-only techniques, we take a joint encoding-decoding optimization approach. Similar to existing methods, our main ideas could be better understood from the binary tree traversal perspective. They are (a) pruning more subtrees, (b) replacing some non-prunable subtrees with other fast-decodable short codes of the same code rates and then prune these “grafted” subtrees, (c) eliminating the remaining non-prunable subtrees by altering their code rates. As seen, both (b) and (c) involve a modified code construction. Consequently, we are able to fast decode any subtree (short code) of a certain size, without sacrificing parallelism.

The algorithmic contributions are summarized below:

1) We introduce four new fast decoding modules for nodes with code rates $\left\{ \frac{2}{M}, \frac{3}{M}, \frac{M-3}{M}, \frac{M-2}{M} \right\}$. Here $M = 2^s$ is the number of leaf nodes in a subtree, where $s$ is the stage number. These nodes are called dual-REP (REP-2), repeated parity check (RPC), parity checked repetition (PCR), dual-SPC (SPC-2) nodes, respectively. More importantly, these modules reuse existing decoding circuits for repetition (REP) and single parity check (SPC) nodes.

2) For medium-code-rate nodes that do not natively support fast decoding, we graft two extended BCH codes to replace the original outer polar codes. BCH codes enjoy good minimum distance and natively support efficient hard-input decoding algorithms, thus strike a good balance between performance and latency. The extension method is also customized to enhance performance.
3) We propose to re-allocate the code rates globally, such that all nodes up to a certain size support the above mentioned fast decoding algorithms. This approach completely avoids the traversal into certain “slow” nodes.

For code length $N = 1024$ and code rate $R = 0.875$, the proposed fast polar codes enable parallel decoding of all length-16 nodes. The proposed decoding algorithm reduces 55% node visits and 43.5% edge visits from the original polar codes, with a cost of within 0.3$dB$ performance loss. Two types of decoder hardware are designed to evaluate the area efficiency and energy efficiency.

The implementation-wise contributions are summarized below:

1) We design a recursive decoder to flexibly support any code rates and code lengths $N \leq 1024$. We estimate that this decoder layout area is only 0.045$mm^2$. For code length $N = 1024$ and code rate $R = 0.875$, it achieves a 25.6$Gbps$ code bit throughput, with an area efficiency of 561$Gbps/mm^2$.

2) We also design an unrolled decoder that only supports one code rate and code length. We estimate that this decoder layout area is 0.3$mm^2$. For code length $N = 1024$ and code rate $R = 0.875$, it provides a 1229$Gbps$ code bit throughput, with an area efficiency of 4096$Gbps/mm^2$.

II. FROM SIMPLIFIED SC DECODING TO FAST POLAR CODES

Following the notations in [6], a node $v$ in a tree is directly connected to a parent node $p_v$, left child node $v_l$ and right child node $v_r$, respectively. The stage of a node $v$ is defined by the number of edges between node $v$ and its nearest leaf node. All leaf nodes are at stage $s = 0$. The set of nodes of the subtree rooted at node $v$ is denoted by $V_v$.

The set of all information bit positions is denoted by $I$ and that of all frozen bits by $I^c$. The set of the information bit positions in subtree $V_v$ is denoted by $I_v$ and the remaining frozen bit positions therein by $I^c_v$.

A leaf node $v_{leaf}$ has no child node, and a root node $v_{root}$ has no parent node.

A. Simplified SC Decoding

If $T^c_v$ matches patterns, a so-called pattern-based simplified decoding can be triggered to process the node in parallel rather than bit-by-bit. From the binary tree traversal perspective, all the child nodes of $v$ do not need to be traversed. Thus decoding latency is reduced.

The existing so-called pattern-based simplified decoding includes 4 different types. A node $v$ is a Rate-1 node [6] if all leaves in the subtree $V_v$ are information bits, and a Rate-0 node [6] if all leaves in the subtree $V_v$ are frozen bits. To improve the decoder’s efficiency, [10] defines single parity check (SPC) and repetition (REP) nodes. We can employ pattern-specific parallel processing for each type of nodes. Obviously, we need to identify and exploit more special nodes or patterns for latency reduction.

In this paper, we present four new types of corresponding nodes which include $2/3/\{M-3\}/\{M-2\}$ information bits, and describe their corresponding fast decoding methods in Section III.

Pattern-based simplified decoding skips the traversal of certain subtrees when it matches the above patterns.

Currently, there are eight pattern types to cover eight code rates of a sub tree: $\{0, \frac{1}{M}, \frac{2}{M}, \frac{3}{M}, \frac{M-3}{M}, \frac{M-2}{M}, \frac{M-1}{M}, 1\}$. In other words, nodes with other code rates cannot be fast decoded. We need to work on the following two parameters.

1) Ratio of simplified nodes: currently eight out of the $M+1$ code rates support simplified decoding. The ratio is thus $\frac{8}{M+1}$. Note that only the lowest and highest codes can be simplified, meaning code rates between $\frac{3}{M}$ and $\frac{M-3}{M}$ do not benefit from the fast decoding algorithm. For short and medium length codes, many nodes fall into this range due to insufficient polarization. We hope to further reduce latency by introducing more fast-decodable patterns to cover more code rates.

2) Degree of parallelism: it can be represented by $M$, since the $M$ bits in a simplified node are decoded in parallel. The larger $M$ is, a larger proportion of the binary tree can be pruned due to simplified decoding. we hope to increase $M$ for higher throughput as well.

For $M = 8$, the ratio of simplified nodes is $8/9$, with only one code rate $\frac{4}{8}$ unsupported, but the degree of parallelism is only $8$. For $M = 16$, the ratio of simplified nodes reduces to $8/17$, leaving a wide gap of nine unsupported code rates $\frac{4}{16}, \ldots, \frac{12}{16}$, but the degree of parallelism doubles.

B. BCH node

To cover medium code rates, we need to find some patterns which can be fast decoded with good BLER performance. The bad news is, to the best of our knowledge, there exists no parallel decoding method for polar codes with code rates between $\frac{3}{M}$ and $\frac{M-3}{M}$. The good news is that the outer codes represented by a subtree can be replaced by any codes, as shown in many previous works [12] [13] [14]. A good solution is removing the polar nodes with code rate falling into the gap, and grafting a different code that allows fast decoding.
BCH codes are good candidates due to their good minimum-distance property and fast hard-input decoding algorithms. If the error correcting capability is \( t \), it is easy to design BCH codes whose minimum Hamming distance is larger than \( 2 \times t \). This leads to good BLER performance. Meanwhile, the Berlekamp-Massey (BM) algorithm can decode a BCH code with \( t = 1 \) or \( t = 2 \) within a few clock cycles. When grafted to polar codes as fast-decodable nodes, hard decisions are applied to the LLRs from the inner polar codes (parent nodes) before sending to the outer BCH codes (child nodes). Here the BCH codes are called “BCH nodes”.

But BCH codes do not readily solve our problem. They only support a few code rates and code lengths, meaning they cannot cover all the code rates within the gap. For the degree of parallelism \( M = 16 \), the target code length is \( 2^4 \), so the nearest code length of BCH is 15. Meanwhile, BCH codes only support code rates \( \frac{7}{15} \) and \( \frac{11}{15} \) within the gap and the corresponding number of information bits are \( k = 7, k = 11 \).

To overcome the issues, we first extend the code length to 16 bits. Specifically, we add an additional parity-check bits to the BCH codes.

Now that we have grafted two types of BCH nodes, the pattern-based decoding can support 10 code rates. The ratio of simplified nodes increases to 10/17, and the maximum gap reduces to \( \frac{4}{16} \). Figure 2 shows the code rates supported by pattern-based decoding for degree of parallelism \( M = 16 \).

### C. Fast polar codes via rate re-allocation

Even with the inclusion of BCH nodes, the fast decoding algorithm could not cover all the code rates of length-16 subtrees. As the second part of the solution, we propose to construct fast polar codes to avoid the “slow” nodes, and only use the existing ten patterns. Here “fast” resembles that of fast SC decoding but is achieved by altering the code construction instead of decoding. We show that it greatly reduces decoding latency and increases throughput with only slight performance loss.

The following steps show how to construct fast polar codes only with node patterns of discontinuous code rates:

1) Employ traditional methods such as Gaussian approximation (GA) or polarization weight (PW) to build polar codes with the parameter of code length \( N \) and code rate \( R \).

2) Split all \( N \) synthesized sub-channels to \( N/16 \) segments. Each segment constitutes a 16-bit long block code, or equivalently a subtree with 16 leaf nodes.

3) Find out all “slow” segments which do not match the supported code rates or patterns. Re-allocate the code rates among segments to match the nearest supported code rate or pattern, which has \( K \) information bits.

4) If the number of information bits of the current segment exceeds or fall short of \( K \), we remove or add a few information bits according to reliability. Apply this process to the remaining “slow” segments until all segments become fast-decodable.

The resulting code is coined as “fast polar code”. A detailed description of the construction algorithm for fast polar codes can be found in [1].

Take code length \( N = 1024 \), code rate \( R = 0.875 \) as an example, we count the number of fast-decodable nodes to be visited and edges to be traversed. These numbers provide a good estimate of SC decoding latency [6] [10], and are thus used to compare between the construction proposed in this section and the GA construction in Table I. As seen, the traversed nodes and edges reduce by 45\% and 43.5\%.
It is worth noting that the proposed fast polar code construction algorithm reallocates the code rates of some nodes against their actual capacity derived from channel polarization. This inevitably incurs BLER performance loss. To evaluate the loss, we run simulations and Fig. 3 compares the BLER curves of both constructions under code length $N = 1024$, and code rates $R = \{0.75, 0.8125, 0.875, 0.9375\}$. There is a maximum of 0.3dB loss at BLER $10^{-2}$ between GA polar codes and the fast polar codes when adopting QPSK modulation.

III. FAST DECODING ALGORITHMS

In this section, we describe the algorithms to support fast decoding of the newly defined SPC-2, REP-2, RPC and PCR nodes. For BCH nodes, we employ the classic BM algorithm which takes hard inputs and supports hardware-friendly fast decoding.

Each fast-decodable node $v$ at stage $s$ can be viewed as an outer code of length $M = 2^s$. The code bits of $v$ as an outer code are denoted by $X_v$, with $M$ bits.

A. SPC-2

For a dual-SPC node $v$, we divide its code bits $X_v$ into two groups, $X_v^{even}$ whose indices are even numbers, and $X_v^{odd}$ whose indices are odd numbers. According to the definition of an SPC-2 node, there are two parity-check bits in the subtree $V_v$, and the corresponding parity functions $p[0]$ and $p[1]$ can be written as

$$
\begin{align*}
  p[0] & : \bigoplus x = 0, x \in X_v^{even} \\
  p[1] & : \bigoplus x = 0, x \in X_v^{odd}
\end{align*}
$$

We add the two parity functions to get a parity function $p[2]$: 

$$
\begin{align*}
p[2] = p[0] \oplus p[1] : \bigoplus x = 0, x \in X_v^{even}
\end{align*}
$$

Since the two parity functions $p[1]$ and $p[2]$ involve two disjoint sets of code bits, the decoding of an SPC-2 node can be parallelized to two SPC nodes. Each SPC node inherits half of the elements from $X_v$. We can reuse two SPC decoding modules to fast decode an SPC-2 node.

B. REP-2

For a dual-REP node $v$, we divide its code bits $X_v$ into two groups, $X_v^{even}$ whose indices are even numbers, and $X_v^{odd}$ whose indices are odd numbers. According to the definition of an REP-2 node, there are two information bits in the subtree $V_v$, and the corresponding parity functions $p[0]$ and $p[1]$ can be written as

$$
\begin{align*}
p[0] & : \bigoplus x = 0, x \in X_v^{even} \\
  p[1] & : \bigoplus x = 0, x \in X_v^{odd}
\end{align*}
$$

We add two parity functions $p[3]$ and $p[4]$: 

$$
\begin{align*}
p[3] = p[1] \oplus p[2] : \bigoplus x = 0, x \in X_v^{even} \\
p[4] = p[0] \oplus p[3] : \bigoplus x = 0, x \in X_v^{odd}
\end{align*}
$$

We define $\hat{c}_i = \bigoplus x, x \in X_v^i, i \in \{0, 1, 2, 3\}$. According to the upper parity functions we can easily verify that the following relationship holds:

$$
\hat{c}_0 = \hat{c}_1 = \hat{c}_2 = \hat{c}_3 = 0
$$
or

$$
\hat{c}_0 = \hat{c}_1 = \hat{c}_2 = \hat{c}_3 = 1
$$

where $\hat{c}_0, \hat{c}_1, \hat{c}_2, \hat{c}_3$ are the virtual repeated code bits.

Given the above knowledge, the decoding algorithm for an RPC node at stage $s$ where $s \leq 2$, can be easily derived as Algorithm 1, in which $\text{sig}(\alpha) \triangleq \begin{cases} 0, & \alpha \geq 0 \\ 1, & \alpha < 0 \end{cases}$.

Algorithm 1 Decoding a repeated parity check (RPC) node.

**Input:**

- The received signal $\alpha_v = \{\alpha_{vk}, k = 0 \cdots M - 1\}$

**Output:**

- The codeword to be recovered: $\hat{x} = \{\hat{x}_k, k = 0 \cdots M - 1\}$

1. Initialize: $\Delta_0 = 0, \Delta_1 = 0$
2. Initialize: $\delta_i = \infty, c_i = 0, p_i = 0$ for $i = 0 \cdots 3$
3. Initialize: $\hat{x}_k = \text{sig}(\alpha_{vk})$ for $k = 0 \cdots M - 1$
4. for $i = 0 \cdots 3$ do
5. for $j = 0 \cdots M/4$ do
6. $k = j \times 4 + i$
7. $c_i = c_i \oplus \text{sig}(\alpha_{vk})$
8. if $|\alpha_{vk}| < \delta_i$
9. $p_i = k$
10. $\delta_i = |\alpha_{vk}|$
11. end for
12. if $c_i = 1$
13. $\Delta_0 = \Delta_0 + \delta_i$
14. else
15. $\Delta_1 = \Delta_1 + \delta_i$
16. end for
17. for $i = 0 \cdots 3$ do
18. if $(\Delta_0 > \Delta_1) \cap (c_i = 0) \cup ((\Delta_0 < \Delta_1) \cap (c_i = 1))$
19. $x_{p_i} \leftarrow \hat{x}_{p_i}$
20. end for

According to the definition of an RPC node, there are three parity-check bits in the subtree $V_v$, and the parity functions $p[0], p[1]$ and $p[2]$ can be written as

$$
\begin{align*}
p[0] : \bigoplus x = 0, x \in X_v^0 \cup X_v^1 \cup X_v^2 \cup X_v^3 \\
p[1] : \bigoplus x = 0, x \in X_v^1 \cup X_v^3 \\
p[2] : \bigoplus x = 0, x \in X_v^2 \cup X_v^3
\end{align*}
$$
D. PCR

For a PCR node \( v \), we divide its code bits \( X_v \) into four groups in the same way as in (1). According to the definition of an RPC node, there are three information bits in this node. They are denoted by \( u_{M-3}, u_{M-2} \) and \( u_{M-1} \).

We define \( c_i, i \in \{0,1,2,3\} \) according to the following equation
\[
[c_0 \ c_1 \ c_2 \ c_3] = [0 \ u_{M-3} \ u_{M-2} \ u_{M-1}] \times G_4 \tag{2}
\]

It can be easily verified that \( X_v^0 \) are the repetition of \( c_0 \), \( X_v^1 \) are the repetition of \( c_1 \), \( X_v^2 \) are the repetition of \( c_2 \), and \( X_v^3 \) are the repetition of \( c_3 \). Thus, we divide the input signal \( \alpha_v \) into four groups according the indices and combine the input signals within each group into four enhanced signals \( \Delta_i, i \in \{0,1,2,3\} \), as in an REP node.

Equation (2) implies the existence of a virtual single parity check code of rate \( \frac{3}{4} \), with virtual code bits \( c_i, i \in \{0,1,2,3\} \), so we can reuse SPC module to decode it.

IV. HARDWARE IMPLEMENTATION

We designed two types of hardware architectures to verify the performance, area efficiency and energy efficiency.

- **Recursive Decoder**: It supports flexible code length and coding rates of mother code length \( N \) from 32 to 1024 with the power of 2. With rate matching, flexible code length with \( 0 < N \leq 1024 \) and code rate with \( 0 < R \leq 1 \) are supported. The \( f_{+/} \) [15] functions in nodes are processed by single PE (processing element) logic, and one decision module to support all 9 patterns\(^2\). The decoder processes one packet at a time.

- **Unrolled Decoder**: It only supports a fixed code length and code rate. In our architecture we hard code code length \( N = 1024 \), and code rate \( R = 0.875 \). This fully unrolled pipelined design combines exclusive dedicated PEs to process each \( f_{+/} \) function in the binary tree. Same to the decision modules that 21 dedicated node specific logic are implemented to support 21 nodes patterns. With 25 packets simultaneously decoding, thanks to the unrolled fully utilization of processing logic and storage, this decoder provides extreme high throughput with high area efficiency and low decoding energy.

Both the above mentioned decoder implementations adopt successive cancellation algorithm accelerated by pattern-based fast decoding. The maximum degrees of parallelization are 128 for SPC and SPC-2 nodes, and 256 for R1 nodes. All other nodes enjoy a degree of parallelism of 16.

A. Bit quantization

An attractive property of polar codes is that SC decoding works well under low-precision quantization (4 bits to 6 bits). Lower precision quantization is the key to higher throughput, as it effectively reduces implementation area and increases clock frequency.

\(^2\)R0 node is bypassed in SC decoding.

There are two types of quantization numbers, one is for channel LLR and the other is for internal LLR. We first test the case with 6-bit input quantization and 6-bit internal quantization. According to Fig 4, this setting achieves the same performance as floating-point. The second one is 5-bit quantization/5-bit internal quantization. It incurs < 0.1dB loss. Finally, 4-bit input quantization/5-bit internal quantization incurs < 0.2dB loss. In this paper, we evaluate the physical implementation result under 5-bit quantization both input and internal signals to strike a good balance between complexity and throughput.

At the same time, we also compare the BLER performance between the original SPC and parallelized SPC. None of the quantization schemes yields harmful loss.

B. Estimation of Layout Area

We carry out the two FPGA implementations for both the recursive and unrolled architectures and convert the results to the physical implementations.

According the FPGA synthesis results, the recursive decoder has 10170 LUTs and 12772 FFs; meanwhile, the unrolled decoder has 66192 LUTs and 55187 FFs. Both decoders avoid the use of memories, making it easy to convert the FPGA results to ASIC. Converted to 16nm technology, the recursive decoder’s estimated synthesis area and the layout size are 0.032mm\(^2\) and 0.045mm\(^2\), respectively at a clock frequency of 1.00Ghz. The unrolled decoder’s estimated synthesis area and the layout size are 0.17mm\(^2\) and 0.30mm\(^2\), respectively at a clock frequency of 1.20Ghz.

V. KEY PERFORMANCE INDICATORS

The key performance indicators (KPIs) are reported in this section. First of all, we evaluate the area efficiency using equation \( AreaEff (Gbps/mm^2) = \frac{InfoSize \times Latency}{Area \times Latency} \).

The recursive decoder takes 40 clock cycles to decoder one packet under fast polar code construction with code length \( N = 1024 \), and code rate \( R = 0.875 \). Thus the throughput is 25.6Gbps for coded bits, and 22.4Gbps for information bits. Converting to 16nm process, the area efficiency for coded bits is 561Gbps/mm\(^2\). 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
The unrolled decoder takes 25 clock cycles to decoder one packet. It is fully pipelined, meaning a new packet of decoded throughput is thus 1229 Gbps for coded bits, and 1075 Gbps for information bits. Converting to 16nm process, the area efficiency for coded bits is 4096 Gbps/mm².

We further evaluate the power consumption and decoding energy per bit through a 200 packets decoding simulation. The process, voltage and temperature (PVT) condition of the unrolled decoder’s power consumption is 30.9mW, and decoding each bit costs 1.21pJ of energy on average; while the unrolled decoder’s power consumption is 784mW, and decoding each bit costs 0.63pJ of energy on average.

We also compare the decoding throughput, area efficiency and power consumption with several high-throughput decoders in literature, and present the results in Table II. From the KPIs, we conclude that unrolled decoders are more suitable for scenarios requiring extremely high throughput but only support fixed code length and rate; recursive decoders are much smaller, which are better for resource constrained devices, and at the same time provides flexible code rates and lengths - a desirable property for wireless communications.

### VI. CONCLUSIONS

In this paper, we demonstrated fast polar decoders that can meet the ultra-high-throughput demand in the next-generation wireless communication systems. We introduced both recursive architecture and unrolled architecture and provided their respective performance and hardware specifications.

### REFERENCES

[1] J. Tong, X. Wang, Q. Zhang, H. Zhang, R. Li, J. Wang, and W. Tong, “Fast polar codes for terahits-per-second throughput communications,” Available on Arxiv https://arxiv.org/abs/2107.08600

[2] W. Saad, M. Bennis, and M. Chen, “A vision of 6G wireless systems: applications, trends, technologies, and open research problems,” IEEE Network, 2019.

[3] 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.

[4] D. Yin et al., “On the Message Passing Efficiency of Polar and Low-Density Parity-Check Decoders,” IEEE Globecom Workshops (GC Wkshps), Rio de Janeiro, Brazil, 2022, pp. 528-534.

[5] X. Liu, Q. Zhang, F. Qin, J. Tong, H. Zhang, C. Zhao, J. Wang, “A 5.10Gbps decoder ASIC for Polar,” IEEE 15th International Symposium on Wireless Communication Systems (ISWCS), Lisbon, 2018, pp. 1-5.

[6] A. Alamdar-Yazdi and F. Kschischang, “A Simplified Successive-Cancellation Decoder for Polar Codes,” in IEEE Communications Letters, vol. 15, no. 12, pp. 1378-1380, December 2011.

[7] O. Dizdar and E. Arikan, “A High-Throughput Energy-Efficient Implementation of Successive Cancellation Decoder for Polar Codes Using Combinational Logic,” in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 3, pp. 436-447, March 2016.

[8] A. Sral, E. G. Sezer, Y. Ertugrul, O. Arikan and E. Arikan, “Terabits-per-Second Throughput for Polar Codes,” 2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops), 2019, pp. 1-7.

[9] S. A. Hashemi, C. Condo, and W. J. Gross, Fast and flexible successive-cancellation list decoders for polar codes,” IEEE Transactions on Signal Processing, vol. 65, no. 21, pp. 5756-5769, Nov. 2017.

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

[11] Y. Wang, K. Narayanan,” Concatenations of polar codes with outer BCH codes and convolutional codes”, Communication Control and Computing (Allerton) 2014 52nd Annual Allerton Conference on, pp. 813-819, 2014.

[12] H. Saber, I. Marsland, “Design of Generalized Concatenated Codes Based on Polar Codes With Very Short Outer Codes”, Vehicular Technology IEEE Transactions on, vol. 66, no. 4, pp. 3103-3115, 2017.

[13] D. Goldin, D. Burstein, “Performance Bounds of Concatenated Polar Coding Schemes”, Information Theory IEEE Transactions on, vol. 65, no. 11, pp. 7131-7148, 2019.

[14] A. Balatsoukas-Stimming, M. B. Parizi and A. Burg, ”LLR-Based Successive Cancellation List Decoding of Polar Codes,” in IEEE Transactions on Signal Processing, vol. 63, no. 19, pp. 5165-5179, Oct. 2015.

[15] J. Tong, X. Wang, Q. Zhang, H. Zhang, S. Dai, R. Li, and J. Wang, “Toward Terabits-per-second Communications: A High-Throughput Implementation of GN-Coset Codes,” IEEE Wireless Communications and Networking Conference (WCNC), 2021, pp. 1-6.

[16] X. Wang, H. Zhang, R. Li, J. Tong, Y. Ge, and J. Wang, “On the construction of $G_N$-coset codes for parallel decoding,” IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea (South), 2020, pp. 1-6.

[17] X. Wang, J. Tong, H. Zhang, S. Dai, R. Li, and J. Wang, “Toward Terabits-per-second Communications: Low-Complexity Parallel Decoding of $G_N$-Coset Codes,” IEEE Wireless Communications and Networking Conference (WCNC), 2021, pp. 1-5.