Latency and Complexity Analysis of Flexible Semi-Parallel Decoding Architectures for 5G NR Polar Codes

Oualid Mouhoubi, Charbel Abdel Nour, Amer Baghdadi

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 August 2022, accepted 19 September 2022, date of publication 21 October 2022, date of current version 4 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3216292

RESEARCH ARTICLE

Latency and Complexity Analysis of Flexible Semi-Parallel Decoding Architectures for 5G NR Polar Codes

OUALID MOUHOUBI, (Member, IEEE), CHARBEL ABDEL NOUR, (Senior Member, IEEE), AND AMER BAGHDADI, (Senior Member, IEEE)
IMT Atlantique, Lab-STICC, UMR CNRS 6285, 29238 Brest, France

Corresponding author: Oualid Mouhoubi (oualid.mouhoubi@imt-atlantique.fr)

This work was supported in part by the Pracom, Cluster on Advanced Research in Communications, Brest, France.

ABSTRACT Polar codes are one of the most recent additions to the family of forward error correction (FEC) codes, having recently been adopted in the 5G New Radio (NR) standard for the control channel. However, the stringent requirements introduced by the 5G standard in terms of block length and code rate flexibility, along with low end-to-end latency and high error correction performance represent a major challenge for their hardware implementation. In this context, we study the impact of main code and decoder design parameters on the latency, throughput, and the hardware complexity of semi-parallel decoding architectures. The impact of these parameters on the hardware efficiency of semi-parallel architectures is significant. Therefore, we propose two multi-frame decoding approaches that increase the throughput and improve the utilisation rate of the processing units of these architectures. Detailed analytical and logic synthesis results are provided and compared for a large range of values in order to constitute a reference for the implementation of flexible, yet efficient FEC decoders for polar codes.

INDEX TERMS 5G, hardware complexity, latency, list decoding, polar codes, successive-cancellation decoding, throughput.

I. INTRODUCTION

Polar codes are a new class of forward-error correction techniques that have been introduced by Arikan [1]. The 5G New Radio (NR) standard recently adopted them for the uplink and downlink control channels, which caught the interest of industry and academia. The 5G control channel enforces block length and code rate flexibility levels considerably above previously published polar code designs in order to accommodate the tremendous growth in the connectivity and data traffic needs. This high flexibility constraint at the transmitter side, is combined with stringent requirements at the receiver, i.e. polar decoder side. Indeed, low block error rate (BLER) and low hardware complexity, added to a processing throughput and latency of tens of Mbps and tens of μs respectively are requested for the 5G control channel [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Khawaja.

One of the most popular decoding techniques for polar codes is the successive-cancellation (SC) algorithm, which is low complexity and well suited to hardware design. Although sufficient for long polar codes, its error correction performance degrades significantly for medium and short code lengths. For the latter, list-augmented SC decoding (SCL) is necessary to improve BLER. The most likely candidate codeword is chosen as the final estimate once the SCL algorithm has completed its decoding of a list of L candidate codewords [3]. For hardware implementations, however, this results in more latency and chip area usage, and decreased throughput [4].

With the stringent requirements imposed by 5G NR, several efforts have aimed at improving throughput and latency by proposing solutions at both hardware design and algorithmic levels [5], [6], [7], [8], [9], [10]. Nevertheless, it is always difficult to find a good compromise between hardware complexity and key performance metrics, especially when

aiming for a flexible design. While fully parallel unrolled and pipelined architectures [11], [12], [13], [14], [15], [16] yield high throughput, they are very limited in terms of flexibility support. For example, a framework that automatically generates high throughput decoder architectures of polar codes is proposed in [16], which only enables design-time flexibility. In this context, semi-parallel decoder architectures [17], [18], [19], [20] are scalable and provide a wide range of algorithmic and architectural alternatives to investigate and adapt based on the desired performance and hardware constraints placed at the implementation level. They are the main option for a flexible implementation. However, it is not straightforward to select the appropriate number of processing elements and other flexibility parameters and to assess the corresponding impact on the performance metrics of the SCL decoder. Indeed, the study of this impact becomes crucial in order to providing design guidelines for the implementation of 5G NR polar decoders. In this regard, we have recently investigated in a previous conference paper [21] the impact of main code and decoder design parameters on the latency and the hardware complexity of semi-parallel decoding architectures. Our work is extended in this paper by investigating the impact of main code and decoder design parameters on throughput and hardware efficiency. Analysis results show that the impact of these parameters on the hardware efficiency of semi-parallel architectures is significant. Therefore, we propose in this paper two multi-frame decoding approaches that increase the throughput and improve the utilisation rate of the processing units of these architectures. Detailed analytical and logic synthesis results are provided and compared for a large range of values in order to constitute a reference for the implementation of flexible, yet efficient FEC decoders for polar codes.

The rest of this paper is organized as follows. Section II provides a brief overview on polar codes and their decoding algorithms. Section III presents the hardware architectures used to implement polar codes, together with the algorithmic and architectural parameters considered in the proposed study. Section IV provides a detailed analysis of the decoding latency. Hardware complexity results and throughput analysis are discussed in Section V, while hardware efficiency analysis is provided in Section VI. Finally, Section VII concludes the paper.

II. PRELIMINARIES

A. POLAR CODES

This code family applies the concept of channel polarization that leads to subsets of good and bad channels. Information bits are mapped to the K most reliable bit-channels while the remaining bits are set to a known value, usually '0', and represent the frozen set. For a codeword length $N = 2^n, n \geq 1$, a (N, K) polar code is a block code with K input bits and N output bits whose generator matrix G is the $n-$th Kronecker power of matrix $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, i.e., $G_N = F^{\otimes n}$. The encoding process is performed by the matrix multiplication $x = u.G$ where $u = (u_0, u_1, \ldots, u_{N-1})$ stands for the sequence input vector consisting of information bits and frozen bits and $x = (x_0, x_1, \ldots, x_{N-1})$ stands for the encoded vector. With stringent constraints on rate flexibility and low decoding latency, polar codes were chosen in 5G NR to encode the uplink and the downlink control information over the physical uplink control/shared channels (PUCCH/PUSCH) and the physical downlink control/broadcast channels (PDCCH/PBCH). They must therefore accommodate a variety of information block lengths, encoded block lengths, and mother polar code lengths [22].

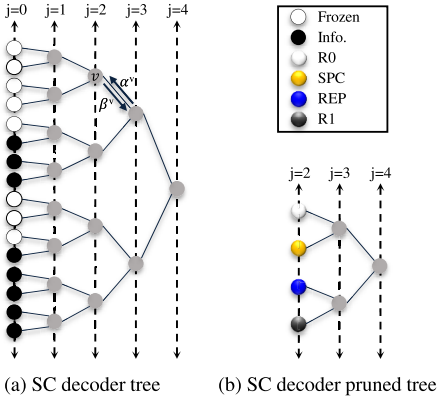


FIGURE 1. SC-based decoder tree and related pruned tree of PC(16,8).

B. DECODING ALGORITHMS BASED ON SUCCESSIVE-CANCELLATION

The decoding of SC algorithm can be performed through a binary tree as illustrated in Fig. 1a for the polar code PC(16,8). It consists of $\log_2 N + 1$ stages where each stage j comprises $\frac{N}{2^j}$ nodes and each node represents a polar code of length 2^j. The top tree node at stage $j = \log_2 N$ includes the channel LLRs (Log-Likelihood-Ratio) and the final Partial Sums (PS). At leaf nodes, the frozen and information bits are represented by white and black circles respectively. A given node v receives α^v LLRs and produces β^v PS. Assuming that the processing of an activated stage j $(0 \leq j \leq n-1)$ can be performed in one time step, the latency required to decode a codeword can be expressed as:

$$\mathbb{L}_{ref} = \sum_{j=0}^{n-1} 2^{n-j} = 2N - 2. \quad (1)$$

The SC algorithm's primary flaw is its inability to correct erroneous bit estimations that arise during the initial stages of decoding. To avoid resorting to hard decisions when computing partial sums during the sequential decoding stage, a SCL technique was devised. On the error-prone bits detected by low reliability values, hard decisions are substituted with soft hypotheses. As a result, few codeword candidates, or equivalently paths in the graph of Fig. 1a, are simultaneously explored, each of which corresponds to one or more different bit-hypotheses. Hence, for the decoding step of each bit u_i, both its possible values 0 and 1 are considered and $2L$ new

candidate paths are explored. However, in order to break the exponential growth in the number of candidate paths, a subset L of the most likely paths is set to survive. The choice is made by selecting the L lowest path metric (PM) values. In terms of complexity, the SCL decoder can be seen as the concatenation of L competing SC decoders. Assuming that a path selection can be performed in one time step, the latency required to decode one codeword with SCL can be expressed as [23]:

$$\mathbb{L}_{\mathrm{SCL}}(N, K) = \mathbb{L}_{ref} + K = 2N + K - 2. \quad (2)$$

A simplified SC-based decoding algorithm (SSC) has been presented in [24] in which the tree search is pruned. Actually, a tree with only frozen bits in its leaves does not need to be traversed since its output is already known and is equal to an all-zero vector. This type of node is called R0. In addition, a tree containing only information bits can be decoded directly by applying a threshold decision to the root node. This type of node is called R1. Furthermore, the Fast-SSC algorithm variant has been the introduced in [25] by identifying two other special types of node among the constituent codes of rate $0 < R < 1$. Hence, a repetition node (REP) is a constituent code where all the bits are frozen except for the last one, and the single parity check (SPC) node is a constituent code where at the exception of the first bit, all the bits are information.

Figure 1b shows the pruned tree of the Fast-SSC decoder, with the four constituent code types being coloured differently. This pruning technique was then extended to the SCL decoder. Therefore, the Simplified SCL (SSCL) in [26] propose an efficient way to decode R0, REP, and R1 nodes of length N_j during at most $\log N_j$, $1 + \log N_j$ and N_j time steps, respectively. This leads to a significant improvement with respect to the conventional SCL algorithm that requires $3N_j - 2$ time steps (assuming adding together N_j values requires at most $\log N_j$ time steps). Furthermore, the SSCL-SPC proposed in [27] provides an efficient decoder for SPC nodes that requires only $N_j + 1$ time steps. However, these two algorithms fail to address the effect of list size on the maximum number of path splits performed at R1 and SPC nodes. This motivated the proposal of Fast-SSCL and Fast-SSCL-SPC algorithms in [28] that can reduce the latency by more than 75% without any degradation in error-correction performance. With these algorithms, the number of time steps required to decode R1 and SPC nodes becomes $\min\left(L - 1, N_j\right)$ and $\min\left(L, N_j\right)+1$, respectively. These numbers can further be reduced with negligible error-correction degradation [28].

III. HARDWARE ARCHITECTURES

Several hardware polar decoder implementations have been developed in the previous ten years for both FPGA and ASIC targets [8], [15], [16], [23], [28], [29], [30], [31]. Unrolled and semi-parallel architectures are the two main architecture models explored in the literature.

A. UNROLLED ARCHITECTURES

The unrolled architecture model consists of assigning a separate hardware resource to each of the operations that occurs during the decoding process. A fully-unrolled and deeply-pipelined architecture is capable of producing one decoded frame per clock cycle by introducing pipeline stages between operations. Therefore, such a decoding architecture can achieve hundreds of Gbps on ASIC technology at the cost of high memory usage. Indeed, a throughput of Tbps is demonstrated in [15] for PC(1024,864) using majority logic aided successive cancellation decoding (SC-MJL) and deeply-pipelined, unrolled hardware architecture with register balancing. However, implementing this form of parallelism results in decoders with high levels of complexity [13]. The main idea behind unrolling a decoder is to increase its throughput. However, this leads to implementations with limited length and rate flexibility, especially when polar codes are decoded using tree-pruning techniques. In fact, a slight change in the location of the information and parity bits within the frozen set of the polar code leads to a whole different list of special node types and sizes. Their suitability for low-latency flexible 5G NR polar decoders is therefore limited.

B. SEMI-PARALLEL ARCHITECTURES

The semi-parallel architecture model integrates a number of processing elements (PEs) devoted to the computation of a single or several types of operations independently of the length or rate of the targeted set of polar codes. When there are more simultaneous operations than there are instantiated PEs, the operations are scheduled into smaller groups and executed sequentially. As a result, the attainable throughput is generally lower than that of the unrolled fully parallel architecture model. Nevertheless, semi-parallel architectures permit the adoption of particular hardware enhancements like memory access sharing and arithmetic resource sharing. This leads to better hardware efficiency as a result of the requirement for flexibility. The SC algorithm's two primary operations, f and g, cannot be overlapped and are always carried out in two separate time periods because of the sequential decoding of SC. Thus, an area-efficient combined processing element is presented in [18] in which these two operations are carried out by a single PE that exploits resource sharing to perform both operations alternately. In fact, semi-parallel decoder architectures are scalable and provide a wide range of algorithmic and architectural alternatives to investigate and adapt based on the desired performance and hardware constraints placed at the implementation level. They are the main option for a flexible implementation. Figure 2 shows the semi-parallel architecture model for SCL decoders used in this work. It includes a set of processing elements, a path selection unit and a partial sum computation unit. The LLRs, partial sums, decoded codewords, and frozen set are all stored in four memory blocks. In addition, the architecture comprises multiplexing networks to interface between memories and computation units. A control unit is used to produce all control signals required during the decoding process. Furthermore, this architecture model may include a unit to decode special nodes when they are considered (dotted block).

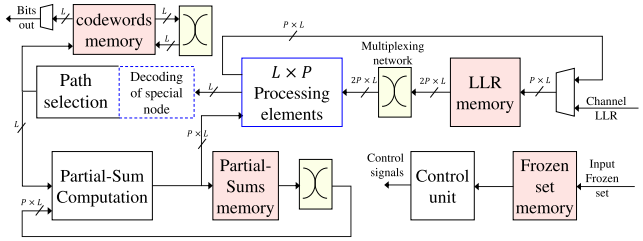


FIGURE 2. Semi-parallel architecture model for SCL decoders.

C. ARCHITECTURAL AND ALGORITHMIC PARAMETERS

The impact of various algorithmic and architectural parameters needs to be investigated while designing low-latency flexible polar decoders with a semi-parallel architecture model. The main parameters that are considered in this work are:

- The number of instantiated PEs: This is a main architectural parameter of the semi-parallel architecture model. Finding the ideal number requires careful analysis with respect to the other flexibility parameters and is not straightforward.
- The tree-pruning techniques: Identifying special nodes in the polar decoder tree followed by applying tree-pruning techniques with corresponding special decoding algorithms can significantly impact the performance metrics of the polar decoder. The influence of different pruning techniques cited in Section II-B are analysed in this work.
- The code length N: A length-flexible implementation increases the complexity of the decoder. Furthermore, the code length may significantly alter the influence of the the above-mentioned parameters on the performance metrics of the polar decoder. For this parameter, the following lengths specified in 5G NR will be considered: $N = 64, 128, 256, 512$ and 1024.
- The code rate R: A rate-flexible implementation decreases the hardware efficiency of the decoding architecture especially when supporting a wide range of values. In this work, we consider the polar codes of PUCCH 5G NR with R ranging from $1/8$ to $5/6$.

One more effective parameter for the latency and throughput is the relationship between the encoder graph and the tree-pruning technique. Indeed, recent works have shown that the polar code can be designed for the tree-pruning technique to achieve low latency [32], [33]. However, since our study is targeting the set of 5G NR polar codes, introducing modifications to the design of the polar codes specified by 3GPP is not an available option to explore. Finally, for certain parameters like data format, quantization, and list size for SCL algorithms, typical values convenient for 5G NR polar codes are taken into consideration. In PEs and special nodes decoders, LLR values are quantized on five and seven bits, respectively, while the list size L is set to eight. LLR values are represented in sign and magnitude (SM) format.

IV. LATENCY ANALYSIS

In this section, we consider analyzing the decoding latency of the 5G NR polar codes in light of the architectural and algorithmic parameters mentioned in the section before. We assume that each elementary operation of the decoding process takes one time-step for the subsequent latency equations and analytical results. As a result, the reported number of clock cycles (CC) required to decode a single frame can be predicted by the number of elementary operations in time-steps, which represents the latency. Using this assumption, the latency of decoding one codeword of length N with K information bits using SC on semi-parallel (SP) architecture can be expressed as:

$$\mathbb{L}_{\text{SC}}^{\text{SP}} = \underbrace{\sum_{j=0}^{p} 2^{n-j}}_{\text{non-affected stages}} + \underbrace{\sum_{j=p+1}^{n-1} 2^{n-j}2^{j-p}}_{\text{affected stages}}. \\ = 2N + \frac{N}{P}\log\left(\frac{N}{4P}\right) \tag{3}$$

where $p = \log_2 P$ and $n = \log_2 N$. This expression is derived from (2) by taking into consideration both affected and non-affected decoding stages j by the introduction of P PEs [18]. For a SCL decoder that comprises P PEs per list, this latency is increased by K since path selection needs to be performed K times [23].

However, any change in the code rate R will have an impact on the latency formula if fast decoding techniques relying on decoding simple constituent codes are employed. Therefore, assuming having the size of all the constituent codes defined for a given N and R, we define E such as $E = \{E_0, E_1, \ldots, E_{\log_2 M - 1}\}$ is the set whose element E_m, $0 \leq m \leq \log_2 M - 1$, represents the number of constituent codes of length 2^{m+1} and M is the size of the largest constituent codes that are considered during the tree-pruning technique. Therefore, to derive the latency of the Pruned Decoder (PD) on a semi-parallel architecture, we should remove the latency of traversing the sub-trees corresponding to the identified special nodes (constituent codes) from the latency of the semi-parallel decoder provided in (3) for $N = 2^n$. This latency reduction can be computed through (1) with $N = 2^{m+1}$ for special nodes that satisfy $m \leq p$, and through (3) with $N = 2^{m+1}$ for the remaining special nodes. The latency needed to decode each of the identified special nodes should also be added, along with K' which is the number of remaining information bits that do not constitute special nodes As a result, the latency of the SCL PD semi-parallel decoder can be stated as follows:

$$\mathbb{L}_{\text{SCL PD}}^{\text{SP}} = \mathbb{L}_{\text{SC}}^{\text{SP}} + \mathbb{L}_{\text{SCL}}^{\text{SN}} + K' \\ - \underbrace{\sum_{m=0}^{p}\left(\sum_{j=0}^{m} E_m \cdot 2^{m-j+1}\right)}_{\mathbb{L}_1}$$

$$
\begin{aligned}
&- \underbrace{\sum_{m=p+1}^{\log_2 M-1}\left(\sum_{j=0}^{p} E_m \cdot 2^{m-j+1}\right)}_{\mathbb{L}_2} \\
&- \underbrace{\sum_{m=p+1}^{\log_2 M-1}\left(\sum_{j=p+1}^{m} E_m \cdot 2^{m-j+1}\cdot 2^{j-p}\right)}_{\mathbb{L}_3}. \\
&= \mathbb{L}_{\mathrm{SC}}^{\mathrm{SP}} + \mathbb{L}_{\mathrm{SCL}}^{\mathrm{SN}} + K' - \sum_{m=0}^{p} E_m \cdot \left(2\cdot 2^{(m+1)} - 2\right) \\
&\quad - \sum_{m=p+1}^{\log_2 M-1} E_m \\
&\quad \cdot \left(2\cdot 2^{(m+1)} + \frac{2^{(m+1)}}{P}\log\left(\frac{2^{(m+1)}}{4P}\right)\right), \qquad (4)
\end{aligned}
$$

where $\mathbb{L}_{\mathrm{SC}}^{\mathrm{SP}}$ and $\mathbb{L}_{\mathrm{SCL}}^{\mathrm{SN}}$ refer to the latency of the SC semi-parallel decoder (3) and the latency required to decode the constituent codes (special nodes), respectively. Also, $\mathbb{L}_1$ is the reduced latency due to constituent codes $\{E_0, E_1, \ldots, E_p\}$ of length smaller or equal to $2^{p+1} = 2P$ while $\mathbb{L}_2$ and $\mathbb{L}_3$ represent the reduced latency due to the presence of constituent codes $\{E_{p+1}, \ldots, E_{\log_2 M-1}\}$ of lengths larger than $2P$.

A. INFLUENCE OF N AND THE NUMBER OF PE ON LATENCY

While algorithmic and architectural parameters are taken into account when designing polar decoders, they are likely to have a significant impact on the latency in the case of semi-parallel architectures. In order to study the influence of N and the number of PE P on the decoding latency, we plot the number of clock cycles required to decode one frame of the 5G NR polar code of four different lengths $N = \{64, 128, 512, 1024\}$ with P varying from 2 to 64. For each value of P, three different algorithms are considered and consist of SCL, SSCL-SPC and the Fast-SSCL-SPC. Based on the speed optimization proposed for the latter [28], three additional values of $\{S_{\mathrm{R1}}, S_{\mathrm{SPC}}\} = \{1, 2\}, \{1, 4\}, \{2, 4\}$ are considered in addition to the optimal variant of this algorithm $\{S_{\mathrm{R1}}, S_{\mathrm{SPC}}\} = \{L-1, L\}$. We refer to them as Fast-SSCL-SPC-12, Fast-SSCL-SPC-14 and Fast-SSCL-SPC-24. S_{R1} is the number of path splits in a R1 node and S_{SPC} is the number of path splits in a SPC node. Simulation results are reported in Fig. 3 while limiting the length of constituent codes to $M = 16$.

As expected, the decoding latency decreases as P increases. However the reduction rate of latency is not linear with respect to P as observed in (3) and this is due to the fact that the degree of parallelism offered by the SC decoder is reduced by half from one higher decoding stage to another lower one. This can be seen in Fig. 3a and Fig. 3d when Fast-SSCL-SPC-12 is used where the latency is reduced by 70% and 52% when P varies from 2 to 8, respectively, while

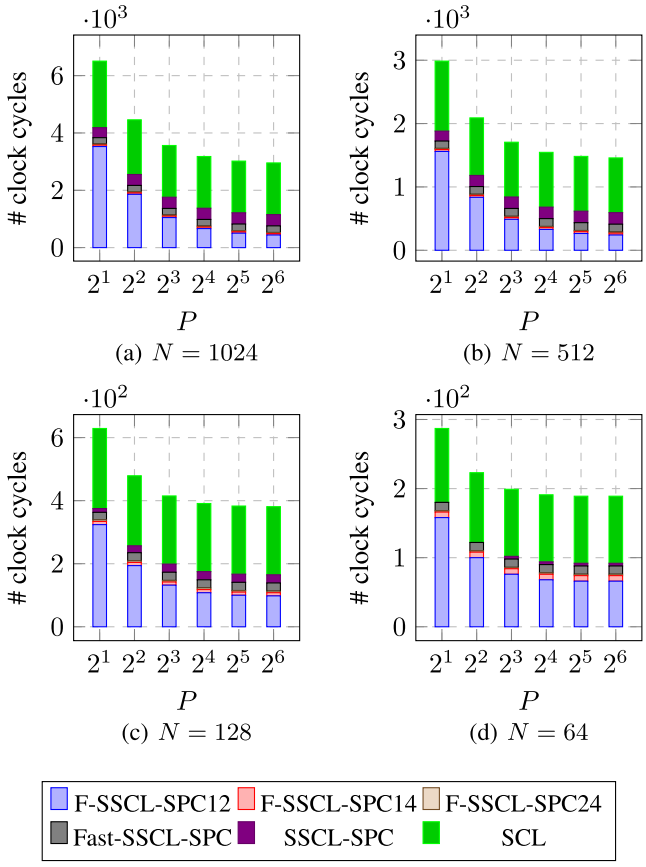


FIGURE 3. Number of clock cycles required to decode one polar code frame for a varying number of PEs. Worst-case latency is reported while varying the value of R. Results are given for SCL and five related variants of simplified algorithms.

it is reduced by a smaller ratio of 58% and 13% when P varies from 8 to 64, respectively. In addition, since a maximum of 32 parallel operations can be completed concurrently, the latency is the same for $P = 64$ and $P = 32$ when $N = 64$. Furthermore, we can clearly observe the impact of the algorithmic choice on the decoding latency. Therefore, using SSCL-SPC instead of SCL with $P = 8$ leads to 44%, 50%, 52% and 49% reduction in latency when $N = 1024$, 512, 128 and 64, respectively. However, a less significant reduction in latency is measured when using Fast-SSCL-SPC instead of SSCL-SPC with $P = 8$. The latency reduction in this case is equal to 22%, 22%, 13% and 4% when $N = 1024, 512, 128$ and 64, respectively. The difference in latency between the various values of N results from the fact that the presence of constituent codes in short polar codes is less important than it is in comparatively long codes. Additionally, when they are identified, their size is usually smaller than $M = 16$.

B. INFLUENCE OF TREE-PRUNING ON LATENCY

Multiple constituent codes of various types and sizes may be present in the decoding tree of polar codes. However, some of these constituent codes, like R0 and REP, are more likely to appear at low code rates, while others, like R1

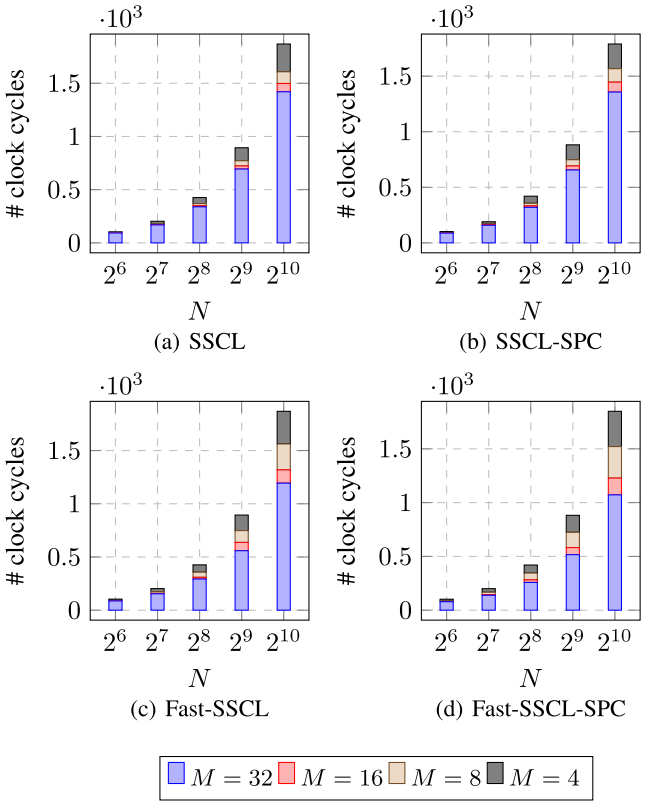


FIGURE 4. Number of clock cycles required to decode one polar code frame as N varies from 64 to 1024. Average latency is reported while varying the value of R. Results are provided for various values of M and are reported for four different algorithms.

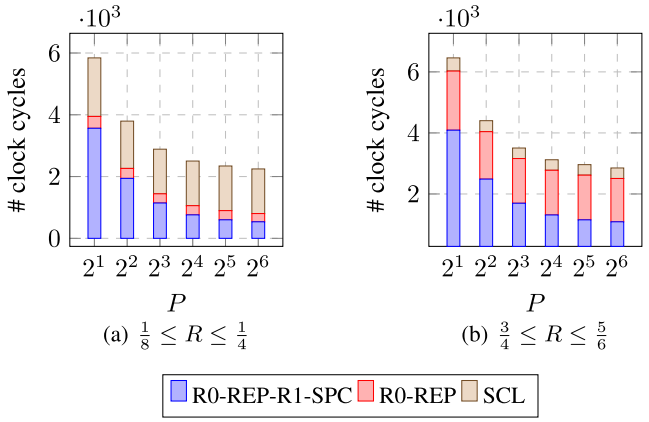


FIGURE 5. Number of clock cycles required to decode one polar code frame for a varying number of PEs and different pruning techniques. Average latency is reported and results are given for low and high code rates.

and SPC, are more likely to appear at high code rates. The requirement for 5G NR to support a wide range of coding rates limits the flexibility available in this regard. On the other hand, large constituent codes are increasingly encountered as the code length increases. To show the impact of M on latency, we plot in Fig. 4 the number of clock cycles required to decode one frame of polar codes of lengths $N = \{64, 128, 256, 512, 1024\}$ for $M = \{4, 8, 16, 32\}$. The same analysis is repeated with SSCL, SSCL-SPC, Fast-SSCL and Fast-SSCL-SPC.

As expected, the number of clock cycles required to decode a constituent code varies according to its type and size. Furthermore, the identification of constituent codes highly depends on M. Some polar codes may benefit better from increasing M than others, especially when they feature large low-latency decoding constituent codes. Indeed, among the set of considered polar codes, the one which achieves the best latency reduction for a fixed value of M does not necessarily produce the same achievement with M', $(M \neq M')$. This implies that using the worst-case latency to evaluate the reduction provided by varying M over a set of rate-variable polar codes that share the same code length leads to an unfair comparison. Consequently, in this analysis we consider the average latency reduction obtained by varying M.

We can see from Fig. 4 that a decoder that can decode constituent codes of size $M = 32$ when targeting polar codes of length $N = 1024$ reduces the latency by 24%, 24%, 36% and 42% in comparison with $M = 4$ under SSCL, SSCL-SPC, FastT-SSCL and Fast-SSCL-SPC, respectively. Nevertheless, a very slight improvement in latency is observed beyond $M = 32$ and does not worth taking into account. Furthermore, we note that setting M to 32 for $N = 64$ does not bring any latency improvement since most polar codes of this length do not have constituent codes larger than $M = 4$.

On the other hand, the influence of tree-pruning on latency also depends on the type of special nodes. As a result of using pruning techniques, the reduction in latency obtained in Fig. 3 and Fig. 4 does not come from the same special nodes. In fact, the latency of decoding polar codes increases as the rate R increases. This observation is true for the SCL algorithm, which requires more path selection operations. Nevertheless, it is also true for simplified SCL algorithms, which feature more R1 and SPC nodes as R increases; hence several clock cycles are required to decode them. Following this observation, The considered worst-case latency reported in Fig. 3 is, in fact, the latency obtained from a polar code whose code rate is close enough or equal to the maximum value of $R \in [\frac{1}{8}, \frac{5}{6}]$. Therefore, this latency reduction comes directly from using R1 and SPC nodes and does not underline the impact of using R0 and REP nodes in reducing the decoding latency. Similarly, the average latency reported in Fig. 4 does not either provide information on the impact of special node types, except the sizes, on latency reduction. Indeed, special node types have a varying impact in reducing the average latency of several polar codes of different code rates. The different types of special nodes do not show an impact in latency reduction with the same proportion when R is varied. Therefore, to underline the impact of special node types in reducing latency, we show in Fig. 5 the average number of clock cycles required to decode one polar code frame at low code rates, $1/8 \leq R \leq 1/4$, and high code rates

$3/4 \leq R \leq 5/6$, distinctively. For this analysis we set M to eight.

As expected, the impact of R0 and REP nodes in reducing the decoding latency is more significant at low code rates. Only a few information bits are present within the frozen set, which leaves space for numerous and large low rate constituent codes, i.e., R0 and REP nodes, to be identified. Therefore, these two special nodes significantly reduce the latency of the conventional SCL algorithm from 30% to 58% for P varying from 2 to 64. However, the impact of these special nodes is less significant at high code rates. Latency is reduced by only 7% to 12% for P varying from 2 to 64. On the other hand, the impact of R1 and SPC nodes in reducing the decoding latency is more significant at high code rates where the polar code features numerous and large special nodes of type R1 and SPC. The latency reduction brought to the decoder by further adding R1 and SPC nodes to the previous SCL algorithm, which already includes the decoding of R0 and REP nodes, is minor for the polar codes of low code rates. At the same time, it is significant for the polar codes of high code rates. It can be observed from Fig. 5 that only 10% to 28% reduction is reported at low code rates against 32% to 57% at high code rates, for P varying from 2 to 64.

V. HARDWARE COMPLEXITY AND THROUGHPUT ANALYSIS

A. INFLUENCE OF THE NUMBER OF PE ON HARDWARE COMPLEXITY

As the latency decreases with increasing P, the complexity of the decoder increases. To analyse the relationship between the complexity of the semi-parallel architecture and P, we propose to implement the processing element of [18] and to estimate the complexity considering the published results for $N = 1024$ as a reference. In [18], two semi-parallel architectures are designed with $P = 16$ and $P = 64$. Using as a reference the design with $P = 16$, and the logic synthesis results obtained from the design of a single PE, we estimated the hardware complexity for $P = 2$, 8, 32, and 64. In this estimation, we simply add and subtract from the reference design the hardware resources (lookup tables and flip-flops) corresponding to the number of PEs, while assuming the remaining components of the decoder unchanged. Figure 6 shows complexity results in terms of lookup tables (LUTs), which are more prevalent in terms of quantity and variation than flip-flops (FFs). A slight inaccuracy in the complexity estimation approach may be seen when comparing the results for $P = 64$, although this was to be expected as part of the design was taken to be constant. However, the relative comparison with respect to several P values offers good insights about how they affect hardware complexity. The results indicate that the impact of the number of PE on complexity is relatively limited. In fact, when P increases by a factor of $\times 32$, from 2 to 64, the overall number of LUTs of the semi-parallel decoder architecture increases by only a factor of $\times 1.55$. On the other hand, this increase in the number of

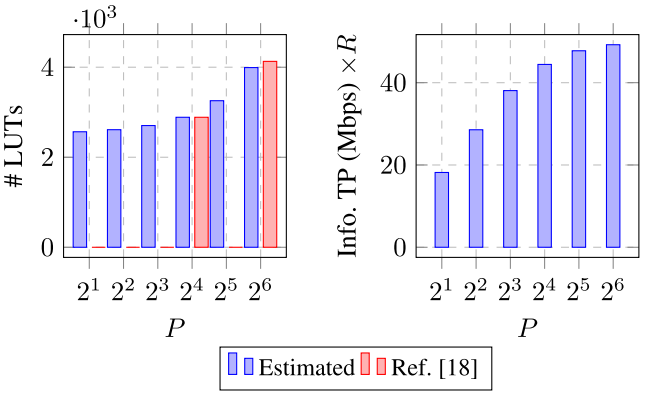


FIGURE 6. Hardware complexity and information throughput as function of P for $N = 1024$. Operating frequency is set to 100 MHz.

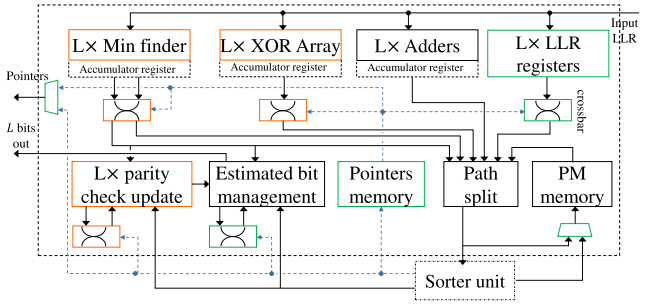


FIGURE 7. Architecture of the SNLD designed to decode special nodes.

PEs leads to an increase in information throughput by a factor of $\times 2.7$ when considering the latency expression provided in (3) and an operating frequency of 100 MHz. It is interesting to notice here that the increase in the information throughput when varying P gradually from 2 to 64 is not linear. It is higher for small values of P. For instance, the throughput increases by 57% when P increases from 2 to 4, whereas it increases by only by 3% when P increases from 32 to 64. This is due to the parallelism bottleneck of SC algorithms. Indeed, as the number of PEs increases, less stages of the decoding tree of polar codes can benefit from the added PEs. In the example of Fig. 6 for $N = 1024$, when $P = 64$ only the highest four stages make a full use of the 64 PEs. However, no change in decoding speed happens for the lowest seven stages when increasing P from 32 to 64. Extended analysis on throughput is provided in Section V-C.

B. INFLUENCE OF TREE PRUNING ON HARDWARE COMPLEXITY

The decoding of constituent codes obtained with tree-pruning techniques requires dedicated hardware resources, beyond those required by the classical SCL semi-parallel decoder. A specific hardware unit namely Special Node List Decoder (SNLD), depicted in Fig. 7, is designed to support decoding the constituent codes R0, REP, R1 and SPC according to SSCL and SSCL-SPC algorithms independently from the PEs. The complexity due to tree-pruning techniques is then

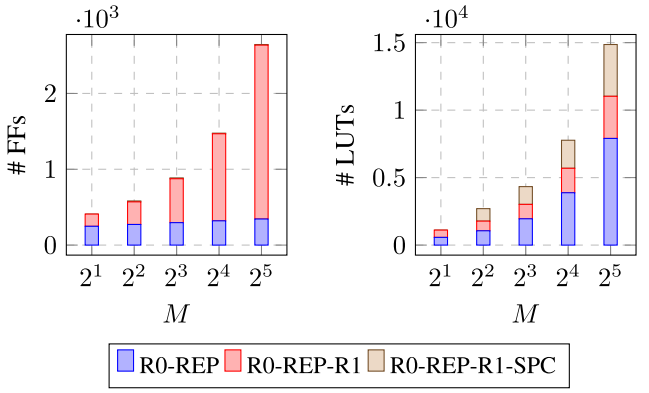


FIGURE 8. Number of FFs and LUTs required by SNLD to decode special nodes R0, REP, R1 and SPC as a function of M according to three scenarios.

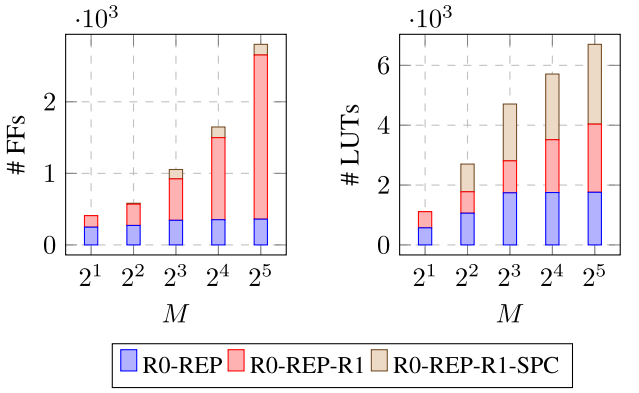


FIGURE 9. Number of FFs and LUTs required by SNLD to decode special nodes R0, REP, R1 and SPC when considering the second approach with accumulator registers to reduce the number of adders, comparators and XOR gates.

evaluated taking into consideration the implementation of the SNLD for $L = 8$.

To do that, the proposed SNLD unit has been described in VHDL and synthesized on a Xilinx Virtex-7 XC7VX-485T FPGA device for three different scenarios. In the first scenario, the SNLD is designed to support only the decoding of R0 and REP nodes. The support of R1 nodes is added in the second scenario through the addition of the green-colored blocks in Fig. 7. The third scenario supports in addition R1 and SPC nodes through the addition of the orange-colored blocks (Fig. 7). SNLD allows resource sharing of the operations that are common to the different special nodes. In the synthesis results, the contribution of the sorter unit has been removed as it is unchanged for all the explored decoding algorithms. The LLR values are represented in SM format, and both LLR and PM values are quantized to 7 bits. For each of the three scenarios, M is varied from 2 to 32 assuming each time that $P = M$ so that the SNLD receives its LLRs in one clock cycle. From the synthesis results presented in Fig. 8, we note that the number of FFs used to decode R0-REP special nodes is almost constant regardless of M. However, the number of FFs starts to increase with M when decoding R1 nodes. This is due to the register array, implemented to store the input LLRs until they are all processed one by one. Since the SC-based decoder does not process multiple nodes simultaneously due to its sequential nature, R1 and SPC nodes do not overlap and the same LLR register array is used store the LLRs of both nodes. Therefore no additional FFs are required during the third scenario that includes further the decoding of SPC nodes. On the other hand, the number of LUTs involved in the first scenario increases linearly with M. This is due to the fact that $M - 1$ adders, designed as a tree-structure fully parallel adder, are implemented to decode R0 and REP nodes. Furthermore, the number of LUTs is increased when R1 nodes start to be considered for $M = 2$ and keep increasing as M increases. This is due to the use of crossbars required to support candidate competition during the successive decoding of the bits of R1 nodes. The decoding of SPC nodes implies performing a parity check equation via a XOR array and searching for a minimum LLR value. Yet, similarly to the first scenario, the complexity of these operations grows linearly with M.

The number of LUTs used by the R0-REP-R1-SPC decoder for $M = 8$ is 3.9 times larger than that for $M = 2$, and that for $M = 32$ is 3.43 times larger than that for $M = 8$. This significant increase in the number of LUTs is mainly due to the growth in the number of used adders and comparators in the minimum finder unit.

In a second strategy, we take into account an implementation based on accumulator register (dotted blocks in Fig. 7) with a smaller tree size for the adders, comparators, and XOR gates. This leads to a semi-parallel architecture of the SNLD with D adders, comparators, and XOR gates, where $D < M - 1$. Figure 9 shows synthesis results for $D = 8$ when using this second approach. When compared to the results of Fig. 8 for the third scenario, a significant improvement can be observed where the number of LUTs is reduced by 26% and 55% for $M = 16$ and $M = 32$, respectively. This comes at the cost of a slight increase in the number of FFs due to the presence of accumulator registers. With this approach, the increase in the number of LUTs when moving from $M = 8$ to $M = 32$ drops from $\times 3.42$ to $\times 1.42$. This significantly reduces the influence of tree pruning on the hardware complexity.

C. INFLUENCE OF PE AND TREE PRUNING ON THROUGHPUT

Due to data dependency between nodes of the SC decoding tree, only a subset of nodes can be activated at a time. Hence, the parallelism level offered by the SC algorithm is limited. The number of operations allowed to be performed in parallel is equal to $N/2$ in the first decoding stage. However, it is continuously halved as we evolve towards leaf nodes, where only a single operation can be performed by visiting one node. Pruning the decoder tree of SC algorithms using direct decoding of specific constituent codes at earlier stages reduces the number of nodes to visit. Also, this increases the exploitable parallelism degrees of SC-based decoders

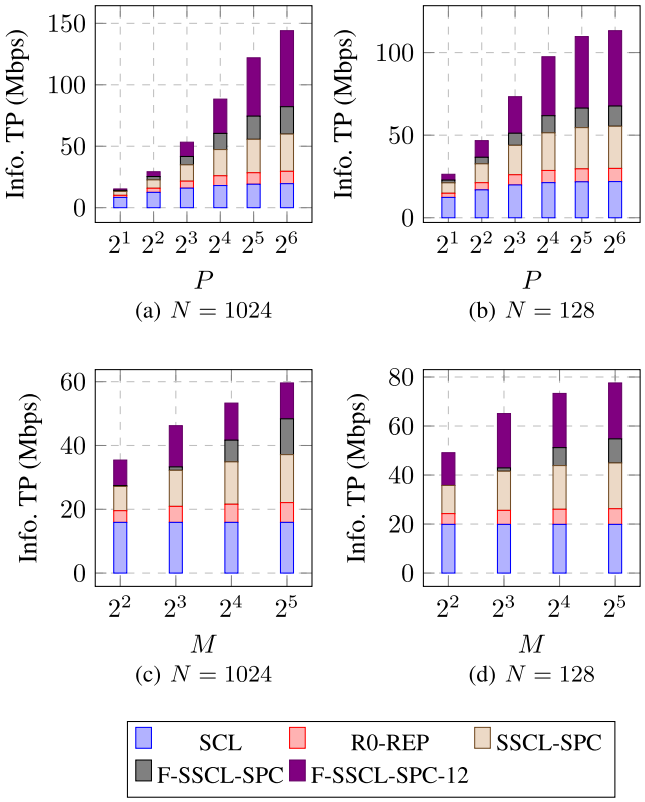


FIGURE 10. Throughput comparison between different decoder types for different values of P and M. Two code lengths are considered $N = 1024$ and $N = 128$.

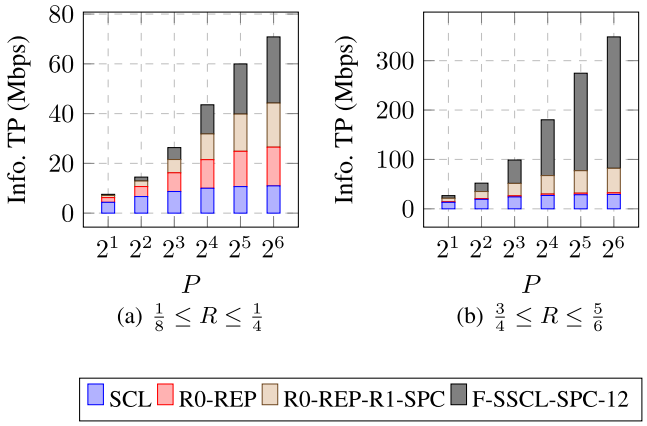


FIGURE 11. Maximum information throughput of SCL decoder and various polar code decoders with different pruning techniques as a function of P. Low and high code rates are considered with $N = 1024$.

since parallel decoding of bits of a special node overcomes data dependency. In this context, we propose to analyze the information throughput of 5G polar codes under the SCL algorithm and fast decoding algorithms, used so far, as a function of P then as a function of M. Throughput is calculated analytically based on the latency equation and assuming an operating frequency of 100 MHz. First, we plot in Fig. 10a and Fig. 10b the average throughput of decoding polar codes of lengths $N = 1024$ and $N = 128$ with $1/8 \leq R \leq 5/6$. The maximum length M of special nodes is set to 16. As discussed earlier, the throughput of SCL decoder increases as P increases. Nevertheless, this increase is not linear when varying P gradually from 2 to 64. Indeed, the throughput of polar codes of lengths $N = 1024$ and $N = 128$ increases by 49% and 37%, respectively, when P increases from 2 to 4, whereas it increases by only 2% and less than 1%, respectively, when P increases from 32 to 64. Furthermore, we can observe from these figures the significant addition to the throughput as soon as fast decoding algorithms relying on tree-pruning are used. For instance, the throughput of the SSCL-SPC decoder, with $P = 8$, is 2.2 times larger than the throughput of the SCL decoder for $N = 1024$. Moreover, It is 3 times higher with a larger P ($P = 64$). By tolerating a loss of 0.1 dB in error correction performance for $N = 1024$, the average achievable throughput with Fast-SSCL-1 ($S_{R1} = 1$) is 144 Mbps, 7.3 times larger than with SCL.

Moreover, decoding multiple bits in parallel overcomes the drawbacks related to the limited exploitable parallelism levels of SCL algorithm. Indeed, fewer nodes are visited to compute LLRs, which reduces the number of access to the stages that do not benefit from increasing P, especially when large and various types of special nodes are considered. Consequently, the increase in throughput tends to be more linear with respect to P compared to SCL.

In order to evaluate the impact of decoding large special nodes on the information throughput, we propose to vary M from 4 to 32 and set P to 8 while using different decoding algorithms. The average throughput of decoding polar codes of lengths $N = 1024$ and $N = 128$ with $1/8 \leq R \leq 5/6$ are reported in Fig. 10c and Fig. 10d. Thus, varying M from 4 to 8 improves the average throughput by 19% and 16% for $N = 1024$ and$N = 128$, respectively. However, varying it from 16 to 32 improves the throughput by only 6% and 2% for $N = 1024$ and $N = 128$, respectively. Furthermore, one can notice that increasing M from 4 to 8 barely improves the throughput of the Fast-SSCL decoder, contrary to when M increases from 8 to 16. In fact, the added value of the Fast-SSCL algorithm compared to SSCL occurs on special nodes of size greater than L. However, this condition is not satisfied in our context, i.e., $L = 8$. Therefore, the Fast-SSCL decoder can enhance the throughput of the SSCL decoder starting from $M = 16$ only.

On the other hand, as shown with latency analysis, the influence of the algorithms based on pruning the decoder tree of polar codes on the throughput depends on the type of special nodes used by the decoder. The large range of rates used in the previous simulation does not reveal the importance of each special node type on improving the throughput. To analyze this effect, we present in Fig. 11a and Fig. 11b the throughput obtained for decoding the polar codes of lengths $N = 1024$ and $N = 128$ at low code rates, $1/8 \leq R \leq 1/4$, and at high code rates $3/4 \leq R \leq 5/6$, respectively. As expected, the impact of R0 and REP special nodes is

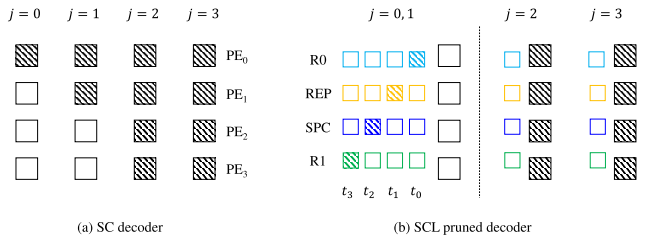


FIGURE 12. Number and type of active (hashed squares) and inactive (blank squares) PEs ($P = 4$) and special node decoders at each stage activation of PC(16,8) of Fig. 1.

more important at low code rates, while it is almost negligible at higher code rates. For instance, these two special nodes increase the throughput of the SCL decoder by 2.15 times and 2.42 times when R is low for $P = 16$ and $P = 64$, respectively. However, they increase the throughput by only 1.03 times and 1.11 times when R is high for $P = 16$ and $P = 64$, respectively. Now, when R1 and SPC nodes are considered in addition to R0 and REP nodes, a significant throughput enhancement is observed for high code rates. Indeed, R1 and SPC nodes increase throughput by 2.25 times and 2.53 times for $P = 16$ and $P = 64$ when SSCL-SPC decoding is considered and by 6 times and 10.72 times when Fast-SSCL-SPC and $\{S_{\text{R1}}, S_{\text{SPC}}\} = \{1, 2\}$. Contrary to R0 and REP nodes which only bring minor throughput improvements at high code rates, the consideration of R1 and SPC nodes still enhances the throughput of the decoder at low code rates. Certainly, this increase is less significant compared to the one observed at high code rates, but quite noticeable considering that the throughput of the SCL decoder is improved by 2.67 times when Fast-SSCL-SPC and $\{S_{\text{R1}}, S_{\text{SPC}}\} = \{1, 2\}$ are considered with 64 PEs.

VI. HARDWARE EFFICIENCY ANALYSIS

The strong data dependency of the SC algorithm limits the performance of the SC decoder in terms of latency and throughput. The solution, based on increasing the number of implemented PEs, becomes less efficient when this number is relatively high compared to the length of the polar code. In addition, the use of a large number of PEs decreases the efficiency of the SC decoders in terms of resource utilization rate. On another note, decoding a group of bits in parallel using tree-pruning algorithms breaks the data dependency of the SC decoder and enhances its performance. To summarize, adding further resources to the conventional decoder does not systematically improve its hardware efficiency.

A. ACTIVITY OF SC DECODERS

To characterize the hardware efficiency of the semi-parallel SC decoder, we propose to analyze the activity of the processing units. We count the number of active processing units (PE, special node decoders) during the time periods spent decoding one polar code frame. Fig. 12a shows the number of active (hashed squares) and inactive (blank squares) PEs at each stage required for the decoding of the PC(16,8) represented by the SC decoder tree of Fig. 1a. The number of PEs used for this example is four. We can notice that all of the implemented PEs are used during the activation of stages $j = 3$ and $j = 2$. However, only half and a quarter of the number of implemented PEs are used during the activation of stages $j = 1$ and $j = 0$, respectively. Given that a stage j is activated 2^{n-j} times, the number of times the semi-parallel SC decoder fully benefits from the PEs when decoding the PC(16,8), considering the additional clock cycles required to update nodes of stages $j = 3$, is equal to $2 \times 2 + 4 \times 1 = 8$. On the other hand, 2 and only 1 of the 4 PEs are used 8 and 16 times respectively at stages $j = 1$ and $j = 0$. This means that the activation of each PE occurs $8 + 8 \times \frac{1}{2} + 16 \times \frac{1}{4} = 16$ times. Therefore, for this example, the activity γ_P of the available PEs is equal to $16 \times P = 64$. For any $p = \log_2 P$ and $n = \log_2 N$, such as $(0 \leq p < n)$, the value of γ_P is expressed as:

$$\gamma_P = \sum_{j=0}^{p} 2^{n-j} \frac{2P}{2^{p-j}} + \sum_{j=p+1}^{n-1} 2^{n-j} 2^{j-p} P. \tag{5}$$

On the contrary, two and three PEs are left unused 8 and 16 times respectively at stages $j = 1$ and $j = 0$ for the example of Fig. 12a. This means that each PE is inactive $8 \times \frac{1}{2} + 16 \times \frac{3}{4} = 16$ times. The inactivity of PEs $\bar{\gamma}_P$ is equal to $16 \times P = 64$ and is expressed as:

$$\bar{\gamma}_P = \sum_{j=0}^{p} 2^{n-j} \left(2 - \frac{1}{2^{p-j}}\right) P + \sum_{j=p+1}^{n-1} 2^{n-j} 2^{j-p} P. \tag{6}$$

To describe the hardware efficiency of the decoder, we use γ_P and $\bar{\gamma}_P$ to derive the utilization rate α. In the case of a SC semi-parallel architecture, $\alpha_{\text{SC}}^{\text{SP}}$ is expressed as:

$$\alpha_{\text{SC}}^{\text{SP}} = \frac{\gamma_P}{\gamma_P + \bar{\gamma}_P}. \tag{7}$$

One can notice that $\alpha_P = 1$ is only reached with one PE ($P = 1$), which means the PE is always active during the decoding process. In the above example, where $N = 16$ and $P = 4$, the utilization rate of the decoder $\alpha_P = 1/2$. Furthermore, if we assume that one PE implementing f and g operations represents twice the complexity of one PE implementing either f or g, (8) results in an exact reformulation of the utilization rate of the SC decoder defined in [18] as the average number of active nodes per clock cycle, which is given by:

$$\alpha_{\text{SC}}^{\text{SP}} = \frac{N \log_2 N}{2P\mathbb{L}_{\text{SC}}^{\text{SP}}}. \tag{8}$$

Following this assumption, the maximum value of the utilization rate is $\alpha_P = 0.5$ for $P = 1$. This is obvious since the implemented PE is always active during decoding. However, this same PE is used alternately to execute f or g functions per node activation. We plot in Fig. 13a the utilization rate as a function of P of the decoders SC, SCL, SSCL, Fast-SSCL, and Fast-SSCL-1 ($S_{\text{R1}} = 1$) of lengths

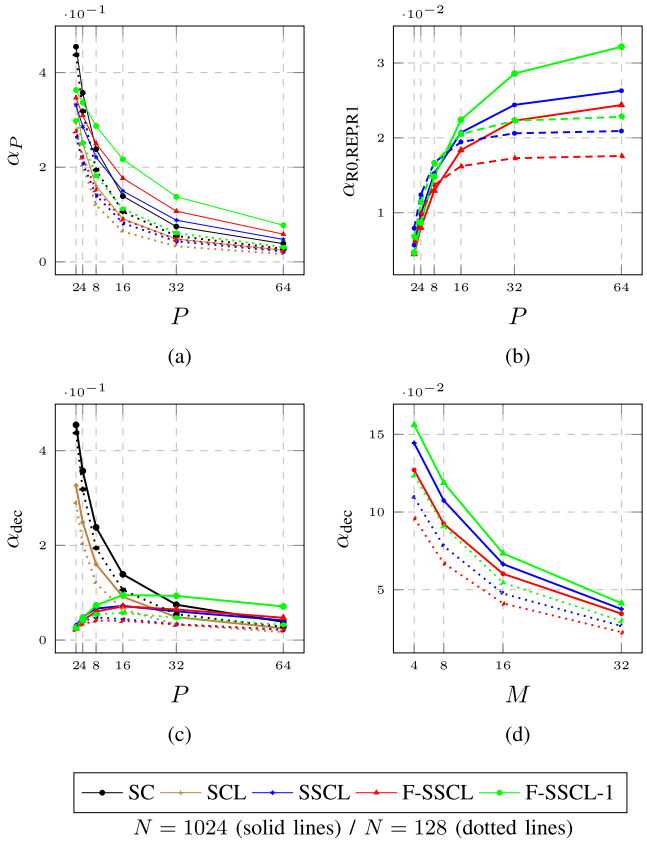


FIGURE 13. Average utilization rates of several SC-based decoders of length $N = 1024$ (solid lines) and $N = 128$ (dotted lines): (a) $\alpha_P(P)$, (b) $\alpha_{SND}(P)$, (c) $\alpha_{dec}(P)$, (d) $\alpha_{dec}(M)$.

$N = 1024$ and $N = 128$. Only the activity of the PEs is studied, while that of the special node decoder is ignored. First, we can see that for the same number of PEs, the α_P of $N = 1024$ is higher than the α_P of $N = 128$. This is because $n - p$, the number of stages where all the P PEs are used, increases with N. Then, it can be noticed that the α_P of the SCL decoder is worse than that of the SC decoder. This is due to the additional decoding clock cycles required to perform candidate competition between different codeword paths when decoding information bits. The value of α_P, in this case, depends on the code rate R. However, for $N = 1024$ and $M = 16$, the SSCL algorithm improves α_P by 1.14 times and 1.85 times for $P = 4$ and 32, respectively, compared to the SCL decoder. This improvement is more significant with the faster Fast-SSCL-1 and results in a 1.34 times and 2.85 times increase in α_P. This is due to the latency reduction acquired by pruning the decoder tree at low decoding stages where a relatively large proportion of the PEs is idle. The PC(16,8) of Fig. 1 is pruned at $j = 2$. Thus, nodes at stages one and zero are not visited anymore during the decoding process. Instead, special node decoders are implemented. The number of active (hashed squares) and inactive (blank squares) PEs together with the special node decoders (colored squares) for R0, REP, SPC, and R1, is shown in Fig. 12b. Since special nodes are decoded in different time periods, their respective decoders are duplicated four times at stages $j = 0, 1$ to show which decoder is active (hashed square) and in what order. We see that all the PEs are used whenever stages $j = 3$ and $j = 2$ are active and idle during special node decoding. In case the Fast-SSCL-SPC-12 decoder is used, the activation of each PE occurs eight times during the 14 clock cycles required to decode one frame, which is higher than the SCL decoder, where it occurs 16 times during 40 clock cycles. The decoding of R0 nodes consists in summing up to M LLR values, and the decoding of REP nodes consists in summing up to M positive LLR values and M negative LLR values, then decoding the information bit. Thus, their computational complexity is equal to M and $2M+1$, respectively. Following this, the utilization rate of α_{R0} and α_{REP} of the example of Fig. 1 in the case of SSCL decoding is $\alpha_{R0} = M/17M$ and $\alpha_{REP} = 2M/(34M + 2)$. Furthermore, the utilization rate $\alpha(Q)$ of an architecture which comprises Q processing units including PEs and various computational complexity special node decoders is expressed as a function of γ_q and $\bar{\gamma}_q$ as:

$$\alpha(Q) = \frac{\sum_{q=1}^{Q} \gamma_q}{\sum_{q=1}^{Q} \gamma_q + \bar{\gamma}_q}, \tag{9}$$

where γ_q and $\bar{\gamma}_q$ are the activity and the inactivity of the processing unit q ($1 \leq q \leq Q$).

We plot in Fig. 13b $\alpha_{R0,REP,R1}$, the average utilization rate of the decoders SSCL, Fast-SSCL, and Fast-SSCL-1 as a function of P while setting M to 16. We consider the activity of the special node decoders of R0, REP, and R1 nodes and ignore the activity of the PEs. As expected, when P increases, the processing of nodes f and g becomes faster, and the activation rate of special node decoders during the decoding period increases, which improves $\alpha_{R0,REP,R1}$. However, this utilization rate is very low compared to α_P presented in Fig. 13a. This is due to the high computation complexity of the special node decoders compared to that of a PE. In addition, the PEs are more used during the decoding process than any of the implemented special node decoders. We also note that, unlike α_P, $\alpha_{R0,REP,R1}$ is better for $N = 1024$ than for $N = 128$, especially for $P \geq 16$. This is because the decoder of length $N = 1024$ benefits better from increasing P beyond $P = 16$ than the decoder of length $N = 128$. On top of that, using special node decoders of size M with the knowledge that polar codes of length $N = 128$ rarely comprise special nodes of this size leads to poor hardware efficiency. Also, we can observe that $\alpha_{R0,REP,R1}$ of the SSCL decoder is better than $\alpha_{R0,REP,R1}$ of the Fast-SSCL. In fact, since Fast-SSCL-1 searches for the minimum LLR out of M LLR values, its computational complexity is M times larger than the SSCL. Hence, its reduced latency benefit is not worth the penalty in terms of hardware efficiency. However, the $\alpha_{R0,REP,R1}$ of the Fast-SSCL-1 decoder is much better than that of the Fast-SSCL since, for the same computational complexity, the Fast-SSCL-1 decoder is much faster. In Fig. 13c we plot α_{dec}, the average utilization rate of the decoders SC, SCL, SSCL, Fast-SSCL and Fast-SSCL-1 ($S_{R1} = 1$) as a function of P for

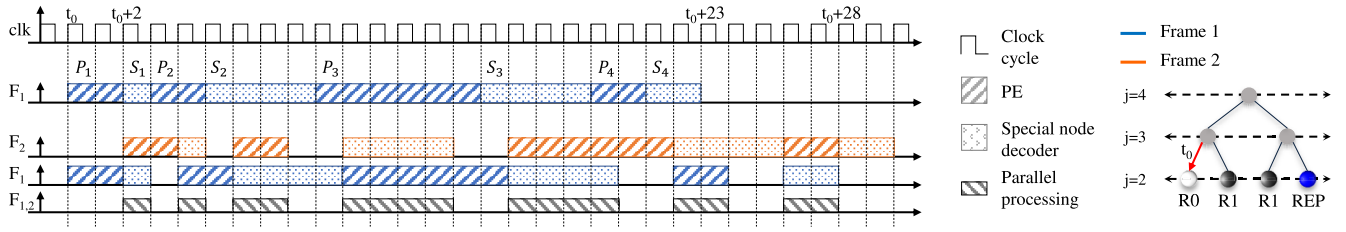


FIGURE 14. Decoding one and two codewords of the same polar code using SSCL algorithm.

$M = 16$ using (9). All the processing units are considered, including PEs and special node decoders of R0, REP, and R1. We can clearly see that for $P < 16$, the SCL algorithm presents a better hardware efficiency than all the simplified algorithms used for this study. When $P < 16$, the number of clock cycles to process non-pruned nodes is very large compared to the number of clock cycles to decode special nodes, i.e., special node decoders are longer idle than active. However, starting from $P = 16$ and $P = 32$ respectively, Fast-SSCL-1 and SSCL become more efficient than the SCL decoder. Using special node decoders increases the decoder throughput, reduces the latency of decoding one polar code frame but decreases the efficiency of the decoder in terms of hardware complexity. Indeed, additional computational units proportional to M are needed to compute the M data inputs related to the top constituent codes nodes. In addition to that, polar codes often include special nodes of variable lengths, which means that a special node decoder designed for M does not frequently take total usage of its hardware resources, especially when M is relatively large. This is shown in Fig. 13d, where the average utilization factor of SSCL, Fast-SSCL and Fast-SSCL-1 decoders is computed for $P = 8$ and various values of M for $N = 1024$ and $N = 128$. It clearly demonstrates that increasing the special node decoder size M reduces the hardware efficiency. Recall that the complexity of one PE is assumed to be twice the complexity of a processing element able to process either f or g operation in the analytical computation of utilization rates of Fig. 13. In addition to that, special node decoders are considered as separate hardware designs. Each decoder has a specific computational resource complexity related to the type and size of the special node. Therefore, when specific hardware optimizations such as arithmetic resource sharing are used, the utilization rate of the decoders is improved.

B. PROPOSED MULTI-FRAME DECODING TECHNIQUES

The poor hardware efficiency of SC decoders is not restricted to semi-parallel SC architectures but also to line and tree architectures [18]. Also pipelined, the tree architecture comprises $N - 1$ PEs, where 2^{j-1} PEs are instantiated at each stage j to perform the available operations whenever the stage is activated. In this way, PEs that belong to non-active stages remain inactive. After duplicating some decoding stages and by adding some PEs, a vector overlapped SC architecture that uses these stages to decode multiple received frames in parallel was presented in [34]. However, memory is also duplicated as many times as the number of vectors decoded in parallel. As a result, this architecture can decode a maximum of $n = \log_2 N$ codewords in parallel without duplicating all the computational resources of the decoder. On the other hand, semi-parallel SC and SCL architectures do not offer many options to overlap the decoding of multiple codewords. The idea of using the idle PEs requires additional hardware complexity to maintain the routing network, but this comes at the cost of a added control complexity. Nevertheless, at the cost of added memory, a semi-parallel decoding architecture similar to the one studied in the previous sections and featuring dedicated special node decoders can be used to enhance the level of parallelism without duplicating computational resources. Precisely, an added memory for holding codeword bits is only required for a length-flexible design when the total length of codewords decoded in parallel does not exceed the designed decoder length. It was shown in Fig. 12 that PEs and special node decoders work alternately during the decoding process. When PEs are used, the special node decoders are not, and vice versa. Therefore, two codewords can be decoded simultaneously with the same hardware resources by alternately updating stages $j = 2, 3$ and $j = 0, 1$. This improves the throughput and hardware efficiency but also increases the decoding latency. Fig. 14 shows the timeline of decoding a sub-tree of one and two received frames of the same polar code using the SSCL algorithm. The polar code selected for this example comprises four consecutive constituent codes of length four, which are R0, two R1s, and REP codes. The number of PEs used by the decoder is $P = 2$. The SSCL decoder alternates between PEs and special node decoders during 23 clock cycles to decode the four special nodes of the sub-tree starting at t_0. Over this time period, PEs are activated 12 times while special node decoders are activated 11 times. The decoding process time is almost equally shared between PEs and special node decoders. This motivates the study of parallel decoding of two frames, F_1 and F_2, using the same architecture. The timeline of decoding F_1 and F_2 shows that these frames can proceed to use the available processing units, i.e., PEs and special node decoders without conflict. However, they cannot use the same type of processing units simultaneously. Therefore, F_1 starts using PEs during two clock cycles denoted by P_1 then uses the special node decoder to decode R0 for one clock cycle denoted by S_1, letting F_2 uses the freed PEs, in turn, for

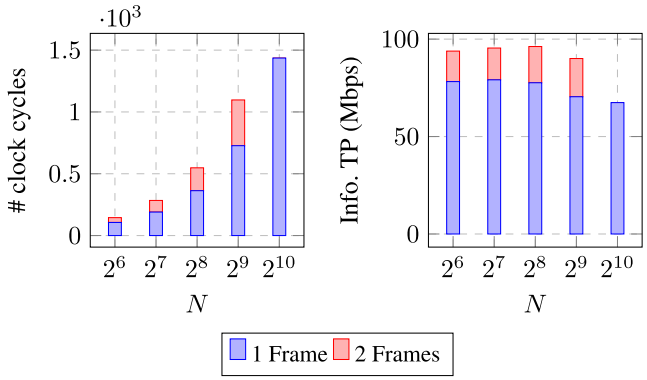


FIGURE 15. Latency and throughput as a function of code length N for a multi-frame SSCL decoder.

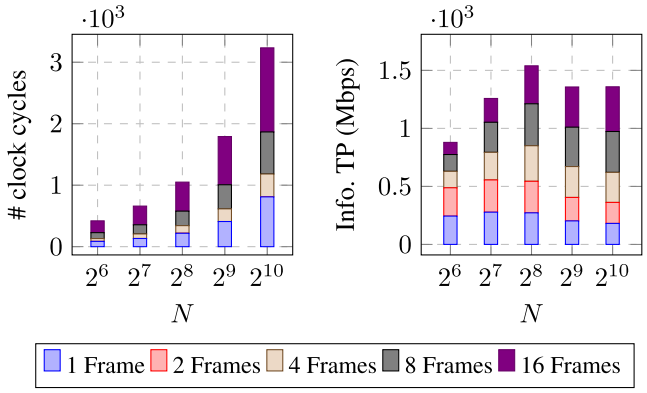


FIGURE 16. Latency and throughput as a function of code length N for a multi-frame Fast-SSCL-1 decoder.

two clock cycles. Nevertheless, since $P_1 > S_1$, F_1 has to wait for $P_1 - S_1$ clock cycles to resume decoding. Similarly, F_2 has to wait for $P_2 - S_1$ clock cycles in order to use PEs the second time around. The remaining special nodes can be similarly decoded with the same schedule and resource sharing process. $F_{1,2}$ highlights the different time periods where both PEs and special node decoders are activated simultaneously by the decoding process of frames F_1 and F_2 (16 clock cycles). The total amount of time required to output the codeword bits of one frame corresponds to 28 clock cycles against 23 clock cycles previously. As a consequence, latency is increased by five clock cycles compared to the case where multiple frame decoding is not applied. The total latency of decoding F frames can be expressed as:

$$\mathbb{L}_M = \sum_q (F-1)\max(P_q, S_q) + \max(P_{q+1}, S_q), \quad (10)$$

where P_q is the number of clock cycles required to process nodes of the pruned decoder tree before decoding the next special node, which requires S_q clock cycles. The average latency and maximum throughput of the 5G polar code over several R values when two frames are decoded in parallel is measured using (10) for $P = 16$ and $M = 16$. The results are provided in Fig. 15 for various polar code lengths. We can see that the latency is increased by 36%, 39%, 43% and 47% while throughput is increased by 20%, 21%, 24% and 28% for $N = 64, 128, 256$ and 512. This increase in latency is more significant compared to the latency analysis conducted above on the simple example of Fig. 14.

Although the overall decoding time is equally shared between PEs and special node decoder, it is frequently uneven between two successive activation of PEs and special node decoders, which results in relatively high decoding latency. For instance, two consecutive 16-length R1 special nodes that share the same parent node lead to increasing the decoder's latency by 15 clock cycles. However, using such a technique guarantees a latency of decoding F frames of length N/F lower than the worst-case latency of decoding one frame of polar codes of length N.

The proposed technique of decoding multiple received frames of the same polar code in parallel presented in this section is based on the notion that the different frames must alternate between using PEs and special nodes with a similar time budget during the decoding process. Therefore, with the same resource, this technique provides good hardware efficiency under the SSCL algorithm but quickly reaches its limits after only two frames are decoded in parallel. However, if faster algorithms are used, particularly the sub-optimal Fast-SSCL, i.e., $S_{\mathrm{R1}} < L - 1$, the time required to decode the set of identified special nodes is much lower than the time required to perform f and g operations.

In this context, we propose another technique that consists in duplicating the PEs as many times as the number of decoded frames in parallel. These multiple frames share the same special node decoder, which increases its utilization rate. Therefore, the decoder architecture includes $F \times P$ PEs and one special node decoder, where F is the number of frames to decode in parallel. The maximum latency and maximum throughput of the Fast-SSCL-1 decoder of lengths $N \in [64, 1024]$ when decoding $F = 1, 2, 4, 8$ and 16 received frames of the same polar code are reported in Fig. 16. The number of PEs and the maximum length of special nodes used in this analysis are $P = 16 \times F$ and $M = 16$, respectively. We see that the latency of decoding two frames is the same as decoding one frame, which is reflected by doubling the throughput of the decoder. A small increase in latency follows increasing F above $F = 2$. Indeed, it reaches an average of 51% and 80% on all the considered code lengths $N \in [641024]$ when increasing F from two to four and eight to sixteen, respectively. This increase in latency is very low compared to the latency required to decode multiple frames when they are decoded one at a time. For instance, in the latter case, the latency of decoding $N = 128$ and $F = 8$ equals 3272 clock cycles while it is equal to 1007 clock cycles (30%) using the proposed technique. Consequently, the throughput of the decoder is significantly enhanced. For instance, it is 3.13, 4.46, and 5.66 times better when decoding four, eight, and sixteen frames of length $N = 256$ in parallel than

decoding a single polar code frame. Nearly similar results are obtained for other code lengths than $N = 256$, as shown in the figure. Far from being limited to $F = 2$, this technique of decoding several received frames in parallel when using the Fast-SSCL decoder provides many advantages in terms of throughput and hardware efficiency at the expense of high memory requirement and duplicated processing elements. Finally, latency and throughput analysis results presented in this section for both multiple frame decoding techniques are performed considering frames from the same polar code, i.e., the same frozen set. Further analysis can be done by considering decoding multiple frames of a variable rate and length.

VII. CONCLUSION

In this paper, we provided a detailed analysis of the impact of main code and decoder design parameters on latency, throughput, hardware complexity and hardware efficiency for polar decoding architectures when targeting the highly flexible 5G NR polar code. In this context, the semi-parallel architecture model proves to be the most suited thanks to a broader algorithmic and architectural flexibility at design level. Therefore, based on a detailed analytical study and logic synthesis results, the latency, throughput and complexity of the decoder were evaluated for multiple variants of fast SCL decoding algorithms and for a varying number of processing elements. Results have shown that length- and rate-flexible designs limit the benefit of increasing the number of processing elements and advocate for defining various types of constituent codes while increasing the level of tree pruning. Indeed, while the use of a large number of processing elements brings a significant latency reduction at large frame sizes, this benefit becomes negligible for short frame sizes, hence penalizing the hardware efficiency of the decoder. Furthermore, some special constituent code types are more likely to appear at low code rates, such that R0 and REP, while others are more likely to appear at high code rates, such as R1 and SPC. Adding to that, the length of these special constituent codes in number of involved bits decreases with the polar code frame size. This can significantly impact implementation efficiency metrics. Therefore, multiple trade-offs between algorithmic and architectural parameters can be drawn from these results. In this regard, we proposed two multi-frame decoding approaches increasing the throughput and improving the processing units activity at the cost of additional memory resources and latency. Finally, the analysis conducted in this paper can be further performed on any other set of polar codes and extended to support list size variation as well.

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, Jul. 2019.

[2] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, "The development, operation and performance of the 5G polar codes," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 96–122, 1st Quart., 2020.

[3] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[4] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 8, pp. 609–613, Aug. 2014.

[5] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency SC polar decoder architectures," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 3471–3475.

[6] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2429–2441, May 2013.

[7] X. Bian, J. Dai, K. Niu, and Z. He, "A low-latency SC polar decoder based on the sequential logic optimization," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.

[8] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 4, pp. 1241–1254, Apr. 2014.

[9] J. S. Roy, G. Lakshminarayanan, and S.-B. Ko, "High-speed architecture for successive cancellation decoder with split-g node block," *IEEE Embedded Syst. Lett.*, vol. 13, no. 3, pp. 118–121, Sep. 2021.

[10] R. Shrestha and A. Sahoo, "High-speed and hardware-efficient successive cancellation polar-decoder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 7, pp. 1144–1148, Jul. 2019.

[11] O. Dizdar and E. Arikan, "A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 3, pp. 436–447, Mar. 2016.

[12] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, "Multi-mode unrolled architectures for polar decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 9, pp. 1443–1453, Sep. 2016.

[13] 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.

[14] C. Kestel, S. Weithoffer, and N. Wehn, "Polar code decoder exploration framework," *Adv. Radio Sci.*, vol. 16, pp. 43–50, Sep. 2018.

[15] A. Sural, E. G. Sezer, Y. Ertugrul, O. Arikan, and E. Arikan, "Terabits-per-second throughput for polar codes," in *Proc. IEEE 30th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC Workshops)*, Sep. 2019, pp. 1–7.

[16] C. Kestel, L. Johannsen, O. Griebel, J. Jimenez, T. Vogt, T. Lehnigk-Emden, and N. Wehn, "A 506 Gbit/s polar successive cancellation list decoder with CRC," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–7.

[17] Y. Delomier, B. L. Gal, J. Crenne, and C. Jego, "Model-based design of hardware SC polar decoders for FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, no. 2, pp. 1–27, Jun. 2020.

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

[19] B. Le Gal, C. Leroux, and C. Jego, "A scalable 3-phase polar decoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 417–420.

[20] A. J. Raymond and W. J. Gross, "Scalable successive-cancellation hardware decoder for polar codes," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Dec. 2013, pp. 1282–1285.

[21] O. Mouhoubi, C. A. Nour, and A. Baghdadi, "On the latency and complexity of semi-parallel decoding architectures for 5G NR polar codes," in *Proc. 11th Int. Symp. Signal, Image, Video Commun. (ISIVC)*, May 2022, pp. 1–6.

[22] *NR; Multiplexing and Channel Coding (Release 15)*, 3GPP, document TS 38.212 V15.2.0, Jan. 2018. [Online]. Available: https://www.3gpp.org/DynaReport/38-series.htm

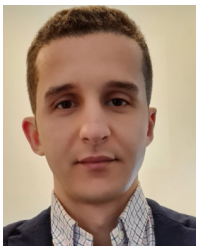
[23] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.

[24] 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.

[25] 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.

[26] S. A. Hashemi, C. Condo, and W. J. Gross, "Simplified successive-cancellation list decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 815–819.

[27] S. A. Hashemi, C. Condo, and W. J. Gross, "A fast polar code list decoder architecture based on sphere decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 12, pp. 2368–2380, Dec. 2016.
[28] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, Nov. 2017.
[29] A. Mishra, A. J. Raymond, L. G. Amaru, G. Sarkis, C. Leroux, P. Meinerzhagen, A. Burg, and W. J. Gross, "A successive cancellation decoder ASIC for a 1024-bit polar code in 180 nm CMOS," in *Proc. IEEE Asian Solid State Circuits Conf. (A-SSCC)*, Dec. 2012, pp. 205–208.
[30] X. Liu, Q. Zhang, P. Qiu, J. Tong, H. Zhang, C. Zhao, and J. Wang, "A 5.16 Gbps decoder ASIC for polar code in 16 nm FinFET," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.
[31] C. Xia, Y. Fan, J. Chen, C.-Y. Tsui, C. Zeng, J. Jin, and B. Li, "An implementation of list successive cancellation decoder with large list size for polar codes," in *Proc. 27th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2017, pp. 1–4.
[32] H. Khoshnevis, C. Cao, D. Chang, and C. Li, "Novel design of irregular polar codes for latency reduction in fast polar decoders," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, Sep. 2021, pp. 1–5.
[33] J. Tong, X. Wang, Q. Zhang, H. Zhang, R. Li, J. Wang, and W. Tong, "Fast polar codes for terabits-per-second throughput communications," 2021, *arXiv:2107.08600*.
[34] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 1665–1668.



OUALID MOUHOUBI (Member, IEEE) received the Engineering degree in electronics and telecommunications from Ecole National Polytechnique, Algiers, Algeria, in 2016, and the M.Sc. degree in embedded systems and information processing from the University of Paris Saclay, France, in 2017. He is currently pursuing the Ph.D. degree with the Department of Mathematical and Electrical Engineering (MEE), IMT Atlantique/Lab-STICC Laboratory, Brest, France. His general research interests include the investigation of efficient hardware architectures for decoding error-correcting codes, and in particular the design of flexible, and low latency decoders for 5G NR polar codes.

CHARBEL ABDEL NOUR (Senior Member, IEEE) received the degree in computer and communications engineering from Lebanese University, Roumieh, Lebanon, in 2002, the master's degree in digital communications from the University of Valenciennes, Valenciennes, France, in 2003, the Ph.D. degree in digital communications from Telecom Bretagne, Brest, France, in 2008, and the Accreditation to Supervise Research degree from the University of Southern Brittany, Lorient, France, in 2020. From June 2007 to October 2011, he was worked as a Postdoctoral Fellow with the Department of Electronics, Telecom Bretagne. He was involved in several research projects related to broadcasting and satellite communications. Additionally during the same period, he was with the Digital Video Broadcasting DVB Consortium, where he had important contributions. Since November 2011, he has been an Associate Professor at Telecom Bretagne and then the IMT Atlantique/Lab-STICC Laboratory. Later, he presented several contributions to the H2020 METIS, FANTASTIC-5G, and EPIC Projects, and the 3GPP consortium related to coding solutions for 5G. His research interests include radio mobile communications systems, broadcasting systems, coded modulations, error correcting codes, resource and power allocation techniques, waveform design, MIMO, and iterative receivers.

AMER BAGHDADI (Senior Member, IEEE) received the Engineering, Master of Science, and Ph.D. degrees from the Grenoble INP (Institut National Polytechnique), France, in 1998, 1998, and 2002, respectively, and the Accreditation to Supervise Research (HDR) in sciences and technologies of information and communication from the University of Southern Brittany, France, in 2012. He is currently a Professor at IMT Atlantique/Lab-STICC Laboratory. His general technical area concerns both algorithm development for digital baseband components and corresponding hardware/software implementations and digital circuit design. His research activities target mainly embedded system design and digital communication applications, in addition to other application domains, and more particularly the design of flexible digital physical layer for future wireless communication standards and terminals. In this context, he has recently contributed to FlexDEC-5G, H2020 METIS, FANTASTIC-5G, and EPIC Research Projects. He serves on the technical program committee of several international conferences.