# On the Latency and Complexity of Semi-Parallel Decoding Architectures for 5G NR Polar Codes

Oualid Mouhoubi, Charbel Abdel Nour, Amer Baghdadi

# On the Latency and Complexity of Semi-Parallel Decoding Architectures for 5G NR Polar Codes

Oualid Mouhoubi[1] Charbel Abdel Nour[1] Amer Baghdadi[1]

[1]IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
(e-mail: firstname.lastname@imt-atlantique.fr)

***Abstract*—Recently adopted in the New Radio (NR) standard for 5G control channel, polar codes represent one of the latest additions to the family of forward error correction (FEC) codes. However, the stringent requirements introduced by the 5G standard in terms of block length and code rate flexibility, together with low end-to-end latency and high error correction performance, pose a major challenge for their hardware implementation. In this context, we study the impact of main code and decoder design parameters on the latency and the hardware complexity of semi-parallel decoding 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, polar codes, successive-cancellation decoding, list decoding, low latency, hardware complexity.**

## I. Introduction

Polar codes are a new class of forward-error correction techniques that have been introduced in 2009 by Arikan [1]. Thanks to their channel capacity-achieving feature and low complexity that have attracted the attention of academia and industry in the past decade, they have been recently adopted in the New Radio (NR) 5G standard for uplink and downlink control channels. They have been first selected for the enhanced mobile broadband (eMBB) service where polar codes with short to moderate block lengths are specified [2]. To meet the tremendous growth in the connectivity and data traffic needs, the 5G control channel imposes block length and code rate flexibility levels far beyond previously published polar code designs. 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 $\mu$s respectively are requested for the 5G control channel [3].

Suitable for hardware design, the low complexity successive-cancellation (SC) algorithm is one of the most common decoding techniques proposed for polar codes. 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. In SCL algorithm, a list of $L$ candidate codewords are considered during decoding [4] and the most likely of them is declared as final estimate at the end of the decoding process. Nonetheless, this comes at the cost of additional latency, chip area occupation and reduction in throughput for hardware implementations [5].

Being the cornerstone for all polar decoder types, several works have targeted the improvement of throughput and latency through the proposal of solutions at both hardware design and algorithmic levels [6]–[11]. However, it is still challenging to find a good trade-off between hardware complexity and key performance metrics particularly when targeting a flexible design. While fully parallel unrolled and pipelined architectures [12]–[15] yield high throughput, they are very limited in terms of flexibility support. In this regard, semi-parallel decoder architectures [16]–[19] are scalable and offer a broad spectrum of algorithmic and architectural options to explore and to adapt depending on the desired performance and hardware limits imposed at the implementation level. This makes them, in turn, a first choice for a flexible implementation. However, selecting the right number of processing elements and assessing its impact, together with other flexibility parameters, on the performance metrics of the SCL decoder is not straightforward. Nevertheless, studying this impact becomes crucial in order to provide design guidelines for the implementation of 5G NR polar decoders. In this regard, we investigate in this paper the impact of main code and decoder design parameters on the latency and the hardware complexity of semi-parallel decoding 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 remainder 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. Latency performance analysis is provided in Section IV while hardware complexity results are discussed in Section V. Finally, Section VI concludes the paper.

## II. Preliminaries

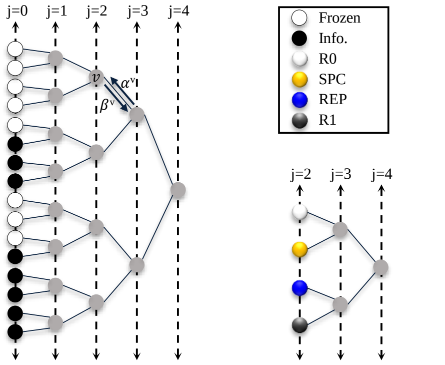
### A. Polar codes

Polar codes concept is based on channel polarization that creates extremal 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). Therefore, they are required to support a wide range of information block length, encoded block and mother polar code lengths [2].

### B. Successive-cancellation decoding based algorithms

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 $\alpha^v$ LLRs and produces $\beta^v$ PS. Assuming that the processing of an activated stage can be performed in one time step, the latency required to decode a codeword can be expressed as:

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



(a) SC decoder tree (b) SC decoder pruned tree

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

The major drawback of the SC algorithm resides in its inability to recover wrong bit estimates when occurring at the early stages of decoding. A SCL algorithm was proposed to avoid resorting to hard decisions when computing partial sums during the sequential decoding phase. Hard decisions are replaced by soft hypotheses on the error-prone bits identified by low reliability values. This leads to the simultaneous exploration of several codeword candidates or equivalently paths in the graph of Fig. 1a, each corresponding to one or more varying bit-hypotheses. Hence, for each bit $u_i$ decoding step, both its possible values 0 and 1 are considered and $2L$ new candidate paths are explored. However, in order to break the exponential growth of 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 [20]:

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

A simplified SC-based decoding algorithm (SSC) was presented in [21] in which the tree search is pruned. In fact, a tree with only frozen bits at leaves does not need to be traversed since its output is already known and is equal to an all-zero vector. This type of nodes is referred to as R0. Moreover, a tree with only information bits can directly be decoded by applying a threshold decision at the root node. This type of nodes is referred to as R1. Furthermore, the authors in [22] have introduced the Fast-SSC algorithm variant by identifying two other special types of nodes 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. The pruned tree of the Fast-SSC decoder is shown in Fig. 1b where the four types of constituent codes are colored differently. This pruning technique was then extended to the SCL decoder. Therefore, the Simplified SCL (SSCL) in [23] 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 significant improvement with respect to the conventional SCL algorithm which 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 [24] provides efficient decoder for SPC nodes which requires only $N_j + 1$ time steps to decode SPC nodes. However, these two algorithms fail to address the effect of list size on the maximum number of path split performed in R1 and SPC nodes. This leads the authors in [25] to propose Fast-SSCL and Fast-SSCL-SPC algorithms which can reduce the latency of more than 75% without any degradation in error-correction performance. With these algorithms, the number of time steps required to decode R1 and SPC becomes $\min(L-1, N_j)$ and $\min(L, N_j)+1$, respectively. These numbers can further be reduced with negligible error-correction degradation.

## III. Hardware architectures

In the last decade, multiple hardware implementations dedicated to decode polar codes have been proposed for both FPGA and ASIC targets [9], [20], [25]–[28] Two main architecture models are investigated in the literature: unrolled and semi-parallel architectures.

### *A. Unrolled architectures*

Unrolled architecture model consists in allocating a dedicated hardware resource for each of the operations that occur during the decoding process. A deeply-pipelined fully-unrolled architecture is able to output one decoded frame at every clock cycle by introducing pipeline stages between the operations. Consequently, such decoder architecture can reach hundreds of Gbps of throughput on ASIC technology at the cost of high memory requirements. However, the quadratic increase with code length of the hardware resources used to support this kind of parallelism leads to a high-complexity decoders [14]. The main idea behind unrolling a decoder is to increase its throughput. However, this results in limited length and rate flexibility implementations, in particular when tree-pruning algorithms are used to decode polar codes. Indeed, bringing a small change to the location of the information and the parity bits withing the frozen set of the polar code leads to a completely different list of special node types and sizes. This limits their suitability for low-latency flexible 5G NR polar decoders.

### *B. Semi-parallel architectures*

Semi-parallel architecture model consists in integrating a certain number of processing elements (PEs) dedicated to compute a single or multiple operation types regardless of the length or the rate of the targeted set of polar codes. In the case where the number of operations to perform in parallel at a given time is greater than the instantiated PEs, these operations are scheduled in sub-groups to be processed sequentially. Therefore, the achievable throughput is typically lower than that of the unrolled fully-parallel architecture model. However, semi-parallel architectures enable the use of specific hardware optimizations such as arithmetic resource sharing and memory access sharing. Considering the flexibility requirement, this leads to improved hardware efficiency. Due to the sequential decoding nature of SC algorithm, its two main operations $f$ and $g$ cannot be overlapped and are always performed in two distinct time periods. Thus, an area-efficient combined processing element is proposed in [17] in which these two operations are carried out by a single PE that exploits resource sharing and can perform both operations alternately. Indeed, semi-parallel decoder architectures are scalable and offer a broad spectrum of algorithmic and architectural options to explore and to adapt depending on the desired performance and hardware limits imposed at the implementation level. This makes them a first choice for a flexible implementation. The semi-parallel architecture model for SCL decoders used in this study is depicted in Fig. 2. It includes a set of processing elements, path selection unit and partial sum computation unit. Four memory blocks store the LLRs, partial sums, decoded codewords and the frozen set. The architecture comprises in addition multiplexing networks to interface between memories and computation units. A control unit is used to produce all control signals required in the decoding process. This architecture model may include in addition a unit to decode special nodes when they are considered (dotted block).

### *C. Architectural and algorithmic parameters*

Targeting a semi-parallel architecture model to design low-latency flexible polar decoders, the impact of several algorithmic and architectural parameters needs to be investigated. The main parameters that are considered in this work are listed below:

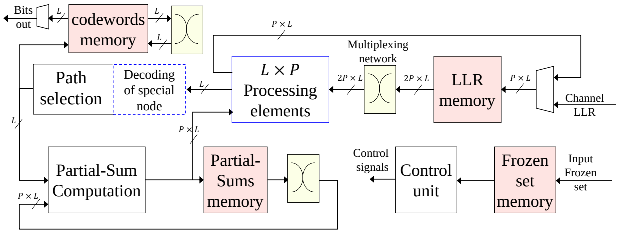


Fig. 2: Semi-parallel architecture model for SCL decoders.

- Number of instantiated PEs: The number of processing elements is a main architectural parameter of the semi-parallel architecture model. Selecting the right number is not straightforward and needs a thorough analysis with respect to the other flexibility parameters.
- Tree-pruning techniques: Identifying special nodes in the polar decoder tree and applying tree-pruning techniques with corresponding special decoding algorithms impact significantly the performance metrics of the polar decoder. The influence of different pruning techniques cited in Section II-B are analysed in this work.
- 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.
- 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 $5/6$.

Typical values, convenient for 5G NR polar codes, are considered for other parameters such as data format, quantization scheme, and list size for SCL algorithms. LLR values are represented in sign and magnitude (SM) format and quantized on five and seven bits in PEs and special nodes decoders, respectively, whereas the list lize $L$ is set to eight.

## IV. Latency analysis

In this section, we propose to evaluate the decoding latency of the 5G NR polar codes under the architectural and algorithmic parameters defined in the previous section. For the upcoming equations of latency and for the analytical results we assume that each elementary operation of the decoding process requires one time-step. Therefore, the reported number of clock cycles (CC) required to decode a single frame corresponds to that number of time-steps, which represents the latency. Following 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:

$$\begin{aligned}\mathbb{L}_{\text{SC}}^{\text{SP}} = & \underbrace{\sum_{j=0}^{p} 2^{n-j}}_{\text{NON-AFFECTED STAGES}} + \underbrace{\sum_{j=p+1}^{n-1} 2^{n-j} 2^{j-p}}_{\text{AFFECTED STAGES}}. \\ = & \, 2N + \frac{N}{P} \log\left(\frac{N}{4P}\right)\end{aligned} \tag{3}$$

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 [17]. For an SCL decoder that comprises $P$ PEs per list, this latency is increased by $K$ as path selection needs to be performed $K$ times [20].

However, if fast decoding techniques relying on decoding simple constituent codes are used, the latency formula will be sensitive to any change of the rate $R$. 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. We should also add the latency that is required to decode each of the identified special nodes and add $K'$ which is the number of remaining information bits that do not constitute special nodes. Thus, the latency of the SCL PD semi-parallel decoder can be expressed as:

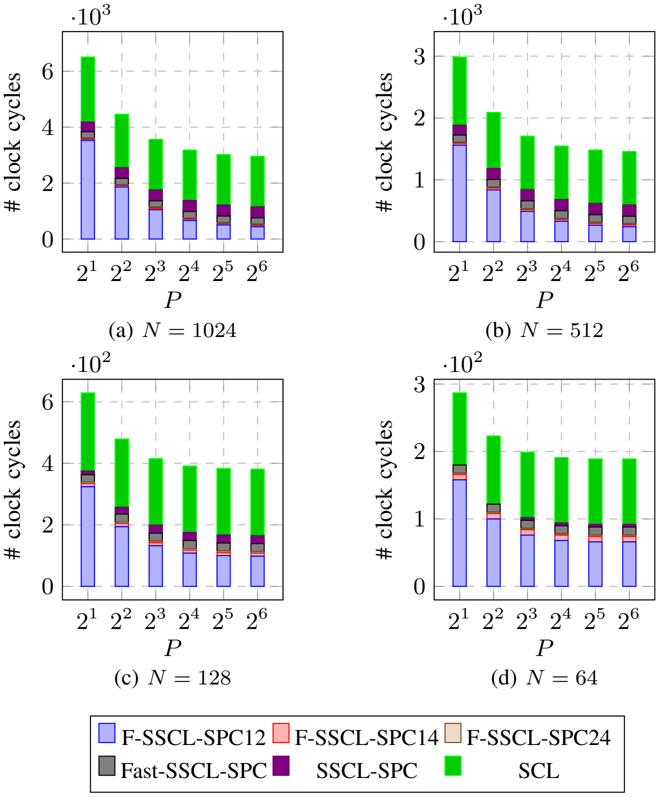
$$\begin{aligned}\mathbb{L}_{\text{SCL PD}}^{\text{SP}} = & \, \mathbb{L}_{\text{SC}}^{\text{SP}} + \mathbb{L}_{\text{SCL}}^{\text{SN}} + K' - \underbrace{\sum_{m=0}^{p}\left(\sum_{j=0}^{m} E_m \cdot 2^{m-j+1}\right)}_{\mathbb{L}_1} \\ & - \underbrace{\sum_{m=p+1}^{\log_2 M-1}\left(\sum_{j=0}^{p} E_m \cdot 2^{m-j+1}\right)}_{\mathbb{L}_2} \\ & - \underbrace{\sum_{m=p+1}^{\log_2 M-1}\left(\sum_{j=p+1}^{m} E_m \cdot 2^{m-j+1} \cdot 2^{j-p}\right)}_{\mathbb{L}_3}. \\ = & \, \mathbb{L}_{\text{SC}}^{\text{SP}} + \mathbb{L}_{\text{SCL}}^{\text{SN}} + K' - \sum_{m=0}^{p} E_m \cdot \left(2 \cdot \mathbf{2}^{(\mathbf{m+1})} - 2\right) \\ & - \sum_{m=p+1}^{\log_2 M-1} E_m \cdot \left(2 \cdot \mathbf{2}^{(\mathbf{m+1})} + \frac{\mathbf{2}^{(\mathbf{m+1})}}{P} \log\left(\frac{\mathbf{2}^{(\mathbf{m+1})}}{4P}\right)\right),\end{aligned} \tag{4}$$

where $\mathbb{L}_{\text{SC}}^{\text{SP}}$ and $\mathbb{L}_{\text{SCL}}^{\text{SN}}$ 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 greater than $2P$.

### A. Influence of N and the number of PE on latency

In the case of semi-parallel architectures, algorithmic and architectural parameters are likely to have a strong impact on the latency when they are considered during the designing of polar decoders. 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 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 [25], three additional values of $\{S_{\text{Rate-1}}, S_{\text{SPC}}\} = \{1, 2\}, \{1, 4\}, \{2, 4\}$ are considered in addition to the optimal variant of this algorithm $\{S_{\text{Rate-1}}, S_{\text{SPC}}\} = \{L-1, L\}$. We refer to them as F-SSCL-SPC12, F-SSCL-SPC14 and F-SSCL-SPC24. $S_{\text{Rate-1}}$ is the number of path split in a R1 node and $S_{\text{SPC}}$ is the number of path split in a SPC node. The 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 F-SSCL-SPC12 is used where the latency is reduced by 70% and 52% when $P$ varies from 2 to 8, respectively, while it is reduced by a smaller ratio of 58% and 13% when $P$ varies from 8 to 64, respectively. In



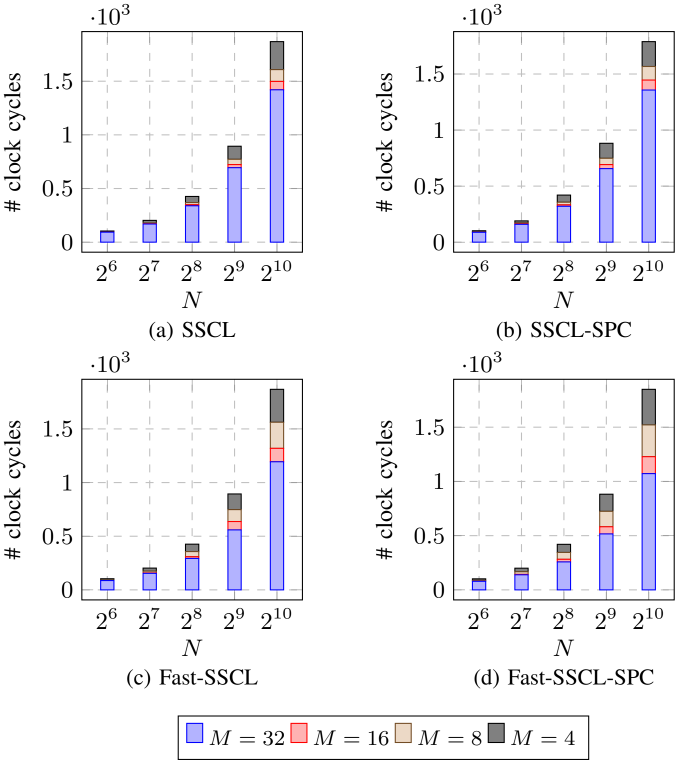
(a) $N = 1024$

(b) $N = 512$

(c) $N = 128$

(d) $N = 64$

Fig. 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.



(a) SSCL

(b) SSCL-SPC

(c) Fast-SSCL

(d) Fast-SSCL-SPC

Fig. 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.

addition to that, the same latency is obtained with $P = 64$ and $P = 32$ when $N = 64$ since a maximum number of 32 parallel operations are granted to be processed in parallel.

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. That disparity in latency between the different values of $N$ is due to the fact that for short polar codes the presence of constituent codes is less significant in comparison to relatively long codes. In addition to that, when they are identified, their size is usually smaller than $M = 16$.

### B. *Influence of tree-pruning on latency*

The decoding tree of polar codes may feature multiple constituent codes of different types and sizes. However, some of these identifiable constituent codes are most likely to appear at low code rates such as R0 and REP while others are most likely to appear at high code rates such as R1 and SPC. Supporting a wide range of code rates, as required in 5G NR, does not offer much flexibility 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 the latency, we plot in Fig. 4 the number of clock cycles required to decode one frame of polar codes of lengths $N \in [64, 1024]$ 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 means that relying on the worst-case latency to evaluate the reduction brought by varying $M$ on a set of rate-variable polar codes that share the same code length leads to an unfair comparison. Therefore, we consider in this analysis the average latency reduction that is obtained as $M$ is varied. 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. However, very small improvement in latency is noticed beyond $M = 32$ and are not worth to be considered. Moreover, we notice that setting $M$ to 32 for $N = 64$ does not bring any improvement in latency since most of the polar codes at this length do not feature constituent codes larger than $M = 4$.

## V. Hardware complexity analysis

### A. *Influence of the number of PE on hardware complexity*

As the latency is decreased with increasing $P$, the complexity of the decoder increases. In order to analyse the relation between the complexity of the semi-parallel architecture and $P$, we propose to implement the processing element of [17] and estimate the complexity considering the published results for $N = 1024$ as a reference. In [17], two semi-parallel architectures are designed with $P = 16$ and $P = 64$. Taking as 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

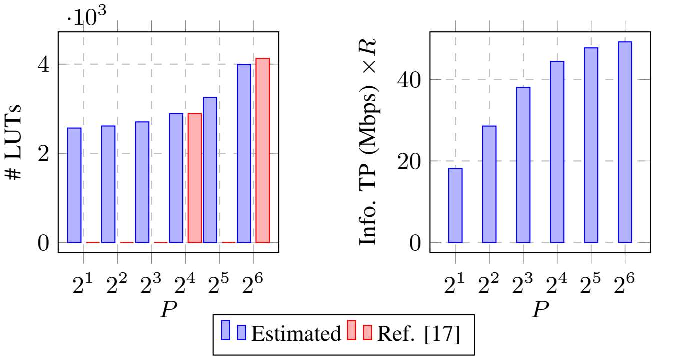


Fig. 5: Hardware complexity and information throughput as function of $P$ for $N = 1024$. Operating frequency is set to 100 MHz.

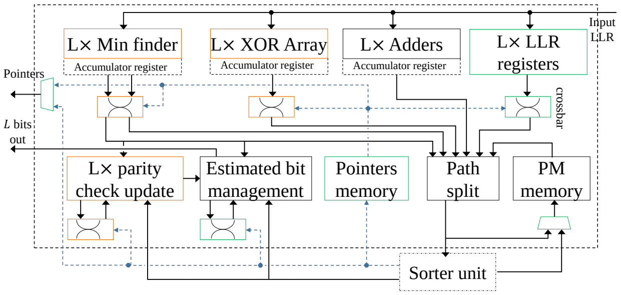


Fig. 6: The architecture of the SNLD designed to decode special nodes.

of PEs, while assuming the remaining components of the decoder unchanged. Complexity results in terms of lookup tables (LUTs), which are predominant in quantity and variation compared to flip-flops (FFs), are reported in Fig. 5. Comparing the results for $P = 64$ reveals a slight inaccuracy in the complexity estimation approach, which is expected as part of the design was assumed unchanged. Nevertheless, the relative comparison with respect to different values of $P$ provides good insights on their impact on the hardware complexity. The results show 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 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 PE increases, less stages of the decoding tree of polar codes take benefit from the added PEs. In the example of Fig. 5 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.

### *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. 6, 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 evaluated taking into consideration the implementation of the SNLD for $L = 8$.

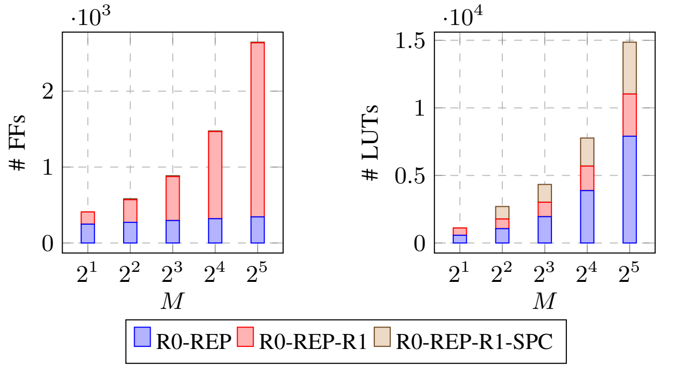


Fig. 7: Number of FFs and LUTs required by SNLD to decode special nodes R0, REP, R1 and SPC as a function of $M$ according to three scenarios.

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. 6. The third scenario supports in addition R1 and SPC nodes through the addition of the orange-colored blocks (Fig. 6). The 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 sign and magnitude (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. 7, 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 grow linearly with $M$.

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

In a second approach, we consider an accumulator register based implementation (dotted blocks in Fig. 6) with reduced 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 $D < M - 1$. Synthesis results when using this second approach are shown in Fig. 8 for $D = 8$. When compared to the results of Fig. 7 for the third scenario, a significant improvement can be observed where

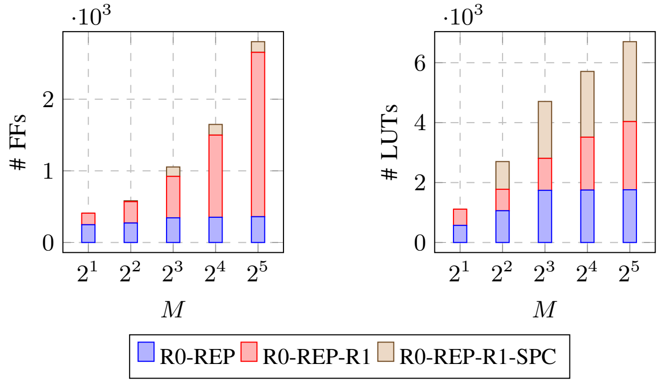


Fig. 8: Number of FFs and LUTs required by SNLD to decode special nodes R0, REP, R1 and SPC when considering the second approach with accumulator registers to reduce the number of adders, comparators and XOR gates.

the number of LUTs is reduced by 26% and 55% for $M = 16$ and $M = 32$, respectively. This comes at the cost of 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.

## VI. Conclusion

In this paper, we proposed a detailed analysis of the impact that main code and decoder design parameters have on the latency and the hardware complexity of polar decoding architectures when targeting the polar codes of the highly flexible 5G NR. For this, the semi-parallel architecture, which offers a broad spectrum of algorithmic and architectural flexibility at design level, is chosen. Therefore, based on a detailed analytical study and logic synthesis results, the latency and the 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, defining various types of constituent codes, and increasing the level of tree pruning. Indeed, while a large number of processing elements brings significant reduction of latency at high code length, the benefit becomes negligible at low code lengths inducing a decrease in the hardware efficiency of the decoder. Furthermore, some 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 codes rates, such as R1 and SPC. Adding to that, large constituent codes may completely disappear at short code lengths. Therefore, multiple trade-offs between algorithmic and architectural parameters can be drawn from these results. 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 Transactions on information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] 3GPP, “TS NR Multiplexing and Channel Coding,” Release 15 V15.6.0 TS, 38.212, 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214

[3] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, “The development, operation and performance of the 5g polar codes,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 96–122, 2019.

[4] I. Tal and A. Vardy, “List decoding of polar codes,” *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[5] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, “Hardware architecture for list successive cancellation decoding of polar codes,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 8, pp. 609–613, 2014.

[6] C. Zhang, B. Yuan, and K. K. Parhi, “Reduced-latency sc polar decoder architectures,” in *2012 IEEE International conference on communications (ICC)*. IEEE, 2012, pp. 3471–3475.

[7] C. Zhang and K. K. Parhi, “Low-latency sequential and overlapped architectures for successive cancellation polar decoder,” *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2429–2441, 2013.

[8] X. Bian, J. Dai, K. Niu, and Z. He, “A low-latency sc polar decoder based on the sequential logic optimization,” in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2018, pp. 1–5.

[9] B. Yuan and K. K. Parhi, “Low-latency successive-cancellation polar decoder architectures using 2-bit decoding,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1241–1254, 2013.

[10] S. J. Roy, G. Lakshminarayanan, and S.-B. Ko, “High speed architecture for successive cancellation decoder with split-g node block,” *IEEE Embedded Systems Letters*, 2020.

[11] R. Shrestha and A. Sahoo, “High-speed and hardware-efficient successive cancellation polar-decoder,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 7, pp. 1144–1148, 2018.

[12] O. Dizdar and E. Arıkan, “A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 3, pp. 436–447, 2016.

[13] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, “Multi-mode unrolled architectures for polar decoders,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 9, pp. 1443–1453, 2016.

[14] ——, “237 gbit/s unrolled hardware polar decoder,” *Electronics Letters*, vol. 51, no. 10, pp. 762–763, 2015.

[15] C. Kestel, S. Weithoffer, and N. Wehn, “Polar code decoder exploration framework,” *Advances in Radio Science*, vol. 16, no. C., pp. 43–50, 2018.

[16] Y. Delomier, B. L. Gal, J. Crenne, and C. Jego, “Model-based design of hardware sc polar decoders for fpgas,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 2, pp. 1–27, 2020.

[17] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, “A Semi-Parallel Successive-Cancellation Decoder for Polar Codes,” *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 289–299, 2013.

[18] B. Le Gal, C. Leroux, and C. Jego, “A scalable 3-phase polar decoder,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 417–420.

[19] A. J. Raymond and W. J. Gross, “Scalable successive-cancellation hardware decoder for polar codes,” in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 1282–1285.

[20] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, “Llr-based successive cancellation list decoding of polar codes,” *IEEE transactions on signal processing*, vol. 63, no. 19, pp. 5165–5179, 2015.

[21] A. Alamdar-Yazdi and F. R. Kschischang, “A simplified successive-cancellation decoder for polar codes,” *IEEE communications letters*, vol. 15, no. 12, pp. 1378–1380, 2011.

[22] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, “Fast polar decoders: Algorithm and implementation,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 946–957, 2014.

[23] S. A. Hashemi, C. Condo, and W. J. Gross, “Simplified successive-cancellation list decoding of polar codes,” in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 815–819.

[24] ——, “A fast polar code list decoder architecture based on sphere decoding,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2368–2380, 2016.

[25] ——, “Fast and flexible successive-cancellation list decoders for polar codes,” *IEEE Transactions on Signal Processing*, vol. 65, no. 21, pp. 5756–5769, 2017.

[26] 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 180nm cmos,” in *2012 IEEE Asian solid state circuits conference (A-SSCC)*. IEEE, 2012, pp. 205–208.

[27] 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 16nm finfet,” in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2018, pp. 1–5.

[28] 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 *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–4.