# Low Latency Architecture Design for Decoding 5G NR Polar Codes

Oualid Mouhoubi, Charbel Abdel Nour, Amer Baghdadi

# Low Latency Architecture Design for Decoding 5G NR Polar Codes

Oualid Mouhoubi, Charbel Abdel Nour, Amer Baghdadi,
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
(e-mail: firstname.lastname@imt-atlantique.fr)

**Abstract.** Polar codes have been adopted as the coding scheme in the control channel of the 3rd Generation Partnership Project (3GPP) New Radio (NR) standard for 5G. However, the challenging requirements introduced by the 5G control channel in terms of block length and code rate flexibility render unsuitable most of the published hardware polar decoder implementations. Indeed, these latter focused mainly on successive cancellation decoders with ultra-high throughput, limited flexibility and error correction capabilities. With stringent constraints on end-to-end delay and error correction, the 5G NR context steers towards low-latency list-based decoder architectures. In this context, we propose an original flexible list-based hardware architecture for decoding 5G NR polar codes that can support all the frame sizes and code rates defined in 3GPP with a worst-case decoding latency below 24 $\mu$s.

**Keywords:** 5G · polar codes · successive-cancellation decoding · list decoding · low latency · hardware design

## 1 Introduction

Proposed in the last few years, Polar codes have been shown to achieve channel capacity in binary discrete memoryless channels, when the codeword length $N$ tends to infinity, using the low complexity successive-cancellation (SC) algorithm [3]. However, performance starts to degrade at practical code lengths. Hence, list-augmented SC decoding (SCL), where a list of $L$ candidate codewords are considered during decoding [19] and the most reliable codeword is chosen at the end, improves the block error rate (BLER). Nonetheless, this comes at the cost of additional latency, chip area occupation and reduction in throughput for hardware implementations. In addition, the BLER of the SCL decoder can be further improved by appending a cyclic redundancy check (CRC) to select the most reliable codeword [14].

Being the cornerstone for all polar decoder types, several works have targeted the improvement of throughput and latency of the SC decoder through the proposal of solutions at both hardware design and algorithmic levels [22, 4, 21, 15, 18]. Through the introduction of specific decoders for constituent codes, simplified successive cancellation decoding (SSC) was proposed in [2] as an alternative solution to reduce the latency. This latest work was then extended to the SCL decoder in [9, 17].

Polar codes have been adopted for the control channel of the enhanced mobile broadband (eMBB) service of the 3GPP's 5G new radio (NR) standard. The control channel imposes block length and code rate flexibility levels far beyond previously published polar code designs. Moreover, low BLER and low hardware complexity added to a processing throughput and latency of 10s of Mbps and 10s of $\mu$s respectively are required [5].

Motivated by the need to provide a hardware-efficient polar decoder that supports the required flexibility and latency levels for 5G NR, we propose in this paper a novel hardware architecture targeting FPGA devices and offering the following features:

– Downlink and uplink 5G NR control channel compliance with full rate and frame size support ranging from $N = 32$ to $N = 1024$ bits.
– CRC-aided SCL decoder with a semi-parallel architecture [10] for best performance.
– Dedicated specific constituent-code decoders for reduced latency.
– Measured FPGA-based worst-case decoding latency of 23.91 $\mu$s, in compliance with target 5G NR constraints.
– Favourable comparison with previously-published designs.

The above-mentioned features were obtained thanks to the following hardware architecture-related contributions:

– Proposal and implementation of on-the-fly identifier for the number of constituent codes in addition to their type and length.
– Introduction of an original special node list decoder capable of decoding all identified constituent-code types.

The remainder of this paper is organized as follows. Section 2 provides a brief overview on polar codes and their decoding algorithms including fast decoding techniques used by our decoder. The proposed on-the-fly identifier and special node list decoder architecture are described in Section 3. Section 4 presents the implementation results and comparisons. Finally, Section 5 concludes the paper.

## 2 Preliminaries

### 2.1 Polar codes

Polar codes apply the channel polarization transform that divides the bit-channels to either perfect or completely noisy channels. Then the polar code allocates information bits to the $K$ most reliable bit-channels while the remaining bits are frozen, i.e., they are all set to a known value, usually '0'. 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.

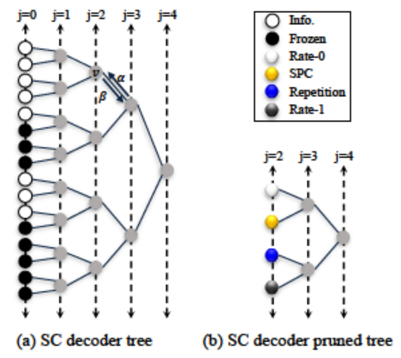

*Fig. 1: SC based decoder tree and its corresponding pruned tree of (16,8) polar code.*

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. CRC bits are appended to the information sequence. Three different CRC generator polynomials were carefully chosen for the purpose of improving the error correction performance. Depending on the physical control channels, CRC are differently initialized, scrambled and interleaved. They are allowed to trigger the end of the decoding process if the CRC check fails particularly during blind decoding. In addition to that, 3GPP decided to integrate in uplink two types of Parity Check (PC) bits in the middle of the encoded block. A so-called universal reliability sequence is used to determine the set of the frozen, information, CRC and PC bit positions for each polar code considered in 5G. With the intention of achieving the desired code rate $R$, reordering the $N$ encoded bits or improving the error correction capability of polar codes, rate matching is the final step introduced in the encoding process used in the 5G NR control channels [1].

## 2.2 Successive-Cancellation decoding based algorithms

The decoding of SC algorithms can be performed through a binary tree as illustrated in Fig. 1a for (16,8) polar code. It consists of $\log_2 N$ stages where each stage $j$ comprises $\frac{N}{2^j}$ nodes and each node represents a polar code of length $2^j$. The top node tree at stage $j = \log_2 N$ includes the channel LLRs 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 clock cycle, the total number of time steps required to decode one frame is: $\mathbb{L}_{ref} = 2N - 2$. This corresponds to $\mathbb{L}_{ref} = 30$ in this example.

The major drawback of the SC algorithm resides in its inability to recover from wrong bit estimates, especially 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 for 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 clock cycle, the latency of the SCL decoder becomes $T_{\text{SCL}}(N, K) = 2N - 2 + K$.

A simplified SC-based decoding algorithm (SSC) was presented in [2] 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 node is referred to as Rate-0. Moreover, a tree with only information bits can directly be decoded by applying a threshold decision at the root node. This type of node is referred to as Rate-1. Furthermore, the authors in[16] have identified two new types of nodes among the constituent codes of rate $0 \leq R \leq 1$. 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. A hardware-friendly PM computation for Rate-0, Rep and Rate-1 nodes and for SPC node is presented in [8] and [7], respectively.

## 3 Proposed Decoder Architecture

### 3.1 Proposed special nodes list decoding module

Instead of decoding every constituent code type individually, a special node list decoder which federates the common operations performed by the different special nodes is proposed. Considering that the tree should be traversed sequentially with a node by node processing schedule, special node decoders are able to share most of the memory and the computational resources. Hence, hardware duplication is avoided. The architecture of the special node list decoder is portrayed in Fig. 2.

Since the Rate-0 nodes comprise only frozen bits, no path splitting is needed to estimate the $N_{\text{Rate-0}}$ bits. Only summing up $N_{\text{Rate-0}}$ values is needed. A tree-structure fully parallel adder is preferred to this purpose in order to guarantee no extra decoding latency. The complexity and the critical path of this adder structure are high especially when the maximum size $M$ defined for special nodes is large. As long as $P \leq M$, where $P$ refers to the number of processing elements

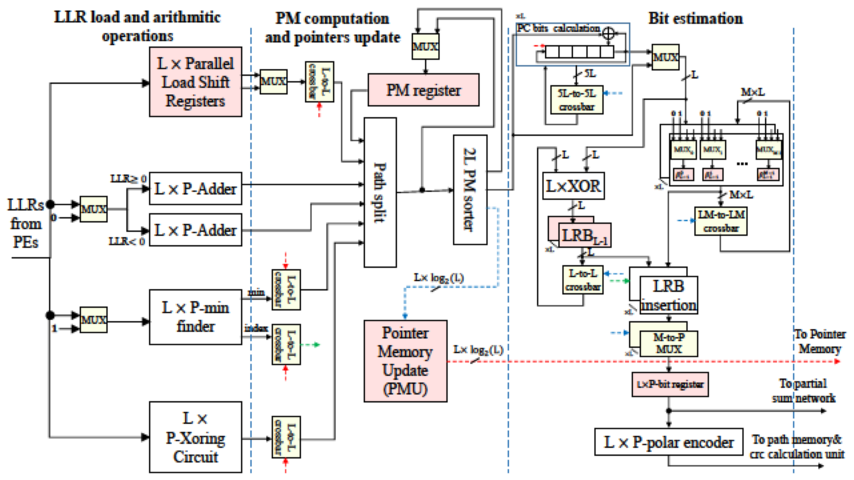

Fig. 2: *Proposed special node decoder architecture supporting different special node types.*

(PE) used by each path, the computations on the special node LLRs take multiple clock cycles and a maximum of $P$ LLRs can be processed at a time. Therefore, smaller tree-structure adder can provide the same computational speed as the fully parallel adder. To this purpose, a sum-accumulate operation is used instead, by introducing an accumulator register at the output of the adder in order to limit the depth of the tree from $\log_2 M$ to $\log_2 P + 1$ stages and reduce the number of adders from $M - 1$ to $P$. As a result, $N_{\text{Rate-0}}$ values can be added up during $\lceil N_{\text{Rate-0}}/P \rceil$ clock cycles, where $\lceil x \rceil$ represents the closest integer value larger than $x$.

A Rep node of length $N_{\text{Rep}}$ is identified by the presence of one information bit and $N_{\text{Rep}} - 1$ frozen bits. However a minimum of two steps are required for the estimate of the single information bit. The first one consists of updating the path metrics as part of the path fork process including both 0 and 1 decisions on the single information bit of the Rep node while the second consists of sorting the PMs of split paths to perform path selection. Unlike $N_{\text{Rate-0}}$ codes, two independent summations are needed by the Rep node. Indeed, according to whether the bit estimation is considered to be 0 or 1, Rep decoder has to add up all the negative valued or positive valued LLRs, respectively. For the purpose of performing both summations in parallel, the Rate-0 node adders are duplicated.

Rate-1 nodes comprise only information bits which are decoded one-by-one as in the SCL algorithm. For each bit estimate, paths are duplicated, sorted and some of them are discarded. Since more than one bit is decoded in a row without going back to PEs, the special node unit comprises a *Pointer Memory Update* (PMU) unit that keeps track of the surviving paths from the beginning of each Rate-1 and SPC nodes decoding.

A SPC node is identified by the presence of one frozen bit while all the remaining bits are information. The least reliable bit is found as a first step for decoding an SPC node. It corresponds to the minimum LLR value at the top of the SPC node tree. An accumulation-minimum-finder tree-based structure similar to the one used for the adder units of Rate-0 and Rep nodes with $P$ comparators is used in this regard. The even-parity check evaluation, in turn, is performed through an accumulator-XOR tree-based structure using $P$ XOR gates, while the evolving even-parity constraint is performed for each of the existing $L$ paths after each bit estimate through $L$ XOR gates and stored in dedicated $LRB$ registers. The final value of the latter is retained to preserve the even-parity constraint and inserted within the estimated bit vectors directly in its appropriate location provided by the minimum value index found in the first step (green arrow in Fig. 2).

Except for a classical bit decoding case, more than one LLR are admitted by the special node decoders. To avoid complexity due to write/read operation, they are stored in a bank of parallel-load shift registers, as illustrated in Fig. 3. Assuming $P \leq M$, they are seen as an array of $M$ $Q_i$-bits registers grouped in $M/P$ independent columns which can be accessed independently thanks to $M/P$ enable signals. $Q_i$ is the quantization level of the internal LLRs. LLR's nodes are stored in the same order they are produced, starting from the rightmost column to the leftmost one during $\left\lceil \frac{2^j}{P} \right\rceil$ clock cycles. During the special node decoding phase, these registers are shifted both horizontally and vertically during the decoding phase of Rate-1 and SPC nodes as follows:

- Vertical shift: LLRs are accessed one-by-one by shifting the rightmost $P$ registers from up to down.
- Horizontal shift: After $P-1$ vertical shift, the following $P$ LLRs to access, when they exist, are obtained by shifting registers of the different columns horizontally from left to right.

Excluding the last loaded registers column, the remaining ones can accept their LLRs from either their adjacent one or from input. This is made possible thanks to $P \cdot \left(\frac{M}{P} - 1\right)$ 2-to-1 MUX. Since the least reliable bit in an SPC node is decoded first, its LLR value should be skipped whenever it is encountered. To do this, the LLR bank register in Fig. 3 is designed to output two adjacent LLR values instead of one. A MUX is used to select the output based on the least reliable bit index. When the PC bits are used in uplink, they are decoded using the length-5 cyclic shift register (Fig. 2). To support path competition, the generated intermediate paths read their values, i.e., LLRs, even-parity check and minimums by means of crossbars. Newly estimated bits are temporarily stored in dedicated registers. A straightforward copy operation allows, by means of crossbars, to move all the contents of each of the $L$ surviving paths from a register to another after each path selection. The process of PM sorting and bit estimation is repeated $N_{\text{Rate-1}}$ or $N_{\text{SPC}} - 1$ times. The PM register is updated directly from the $2L$ sorter in the same clock cycle. When all bits are estimated at their top tree, the source word bits are obtained through a polar encoder, the PMU updates the pointer memory and the search procedure of the next nodes in the polar code tree resumes. The partial sums needed to update LLRs are computed and the CRC check process is resumed. In order to avoid extra latency, PS and LLR computations are performed in parallel.

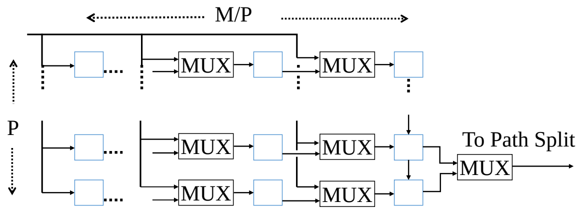

*Fig. 3: Parallel-load shift registers of the top-node LLRs.*

### 3.2 Memory structure

**LLR Memory:** The SC decoding relies on dedicated memories to keep the LLR values available for the computation units as long as they are needed. Therefore, channel input LLRs are stored in a $N \times Q_c$ bits register, where $Q_c$ is the quantization level of the channel LLRs. However, internal LLRs are stored in a dual-port RAM-based memory configured with one write port and one read port. The total number of LLR updates at decoding stage $j$ is equal to $2^j$. However, only $P$ LLR updates are allowed to be performed at once. Thereby, for stages $j$ where $2^j > 2P$, a total of $2^j/(2P)$ time steps are needed before proceeding to the lower stage while only one time step is needed for the other stages. Given that the computed LLRs in PEs have to follow the bit-reversed indexing scheme, the produced LLRs need a certain reordering before being mapped back to PEs. To avoid the need for multiplexing them, the equivalent operation is directly embodied in the control unit and two RAMs instead of one, each of width $PQ_i$, are implemented for each path $l$. For higher stages, LLRs are first stored in one RAM during half of the $2^j/(2P)$ time period required for computation before storing the remaining produced LLRs during the second half of the time period in the other one. However, for lower stages, where LLRs are produced during one time period, LLRs contiguity is ensured by adding a specific permutation network. In addition, a $L \times PQ_i$-bit buffer is used in order to process these generated LLRs directly during the following clock cycle.

**Pointer Memory:** The SCL decoder can also be seen as $L$ SC decoder cores working in parallel with their own LLRs and memory resources. Nonetheless, the cores may share the same LLRs at some stages due to path competition. Thereby, to avoid copying the shared LLRs, a memory pointer is introduced and stored instead. Hence, each SC decoder core can read its inputs from one of the $L$ RAM memories thanks to a permutation network. The used memory pointer is a $(\log_2 N - 1) \times L$ register array, where register at row $j$ and column $l$ stores a pointer of $\log_2 L$ bits indicating the index of the RAM where LLRs of path $l$ at decoding stage $j$ are stored. The pointers are updated in two different situations. First during path duplication/discarding. Second, to reset them to their initial values when the stages at which they are processed is activated.

**Path Memory:** A $LN$ array register is dedicated to store the values of the $L$ codeword bits after the PMs have been sorted and the surviving paths identified. Being mutually independent, all registers can be accessed simultaneously. This is made possible thanks to a $N$-bit enable vector. When a path $l$ needs to be duplicated, all its so far decoded bits are copied to the registers that have been freed, due to path discarding, by means of $N$ $L$-to-1 MUX. Enable signals are generated by a $\log_2 N$-to-$N$ decoder which takes the currently decoding information bit index as input. $N$ MUXes are used to select between the copy/store operation to perform and $P$-to-$P$ crossbars are used to manage the flexibility in decoding variable special node lengths.

## 3.3 Proposed on-the-fly rate-flexible decoding of polar codes

To apply dedicated special node decoders for the 5G NR polar code, the types and sizes of the nodes within the current frame need to be provided. The large flexibility in frame and code rate ranges leads to a large number of different frozen bit sets. This increases the number of different special nodes, making it difficult to store in memory their number and positions. To tackle this issue, we propose a new method to identify the different special node structures (Rate-0, Rep, SPC and Rate-1) on-the-fly directly in hardware without the need to store any list in memory. Their determination is done from the frozen set and by merging different lower size special nodes into a new special node type of larger size. For hardware optimization purposes, we define a new special node type corresponding to the only case where two bits from the frozen set do not represent any of the particular constituent code listed above, i.e., the sequence $\{$*information* , *frozen*$\}$ and is called *No type* node. Henceforth, a unique binary sequence refers to both *No type* and SPC nodes. However, assuming having an extra information on their size, these two special nodes can not be overlapped since SPC nodes start to be considered from length-4 patterns.

Starting from a given frozen set, the structure of Fig. 4a is capable of determining the different special node types of length-2. For higher length nodes, the structure of Fig. 4b intends to merge different lower size special nodes into a new special node type of larger size based on a sequence vector indicating the type of the nodes to merge. With one difference, a SPC node of length four is obtained

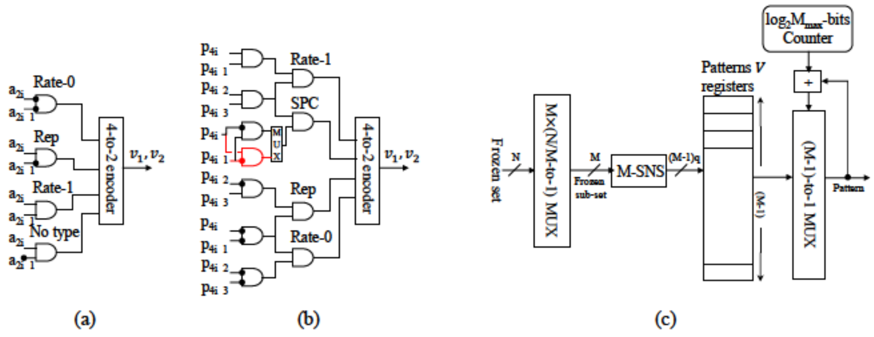

*Fig. 4: Proposed architecture for identification of constituent codes of different lengths: (a) Length two elementary module identifier, (b) Length four and above elementary module identifier, (c) Architecture of the special node identifier.*

by merging a Rep and Rate-1 nodes of length two (highlighted in red) while an SPC node of length greater than four is obtained by merging another SPC and Rate-1 nodes, hence the presence of the MUX. Therefore, the two structures of Fig. 4a and Fig. 4b represent the building blocks of the identifier intended to determine the different supported special node types of any length.

The special and non-special nodes identified are stored together in register arrays in the same order as they are searched during the decoding process. This reduces the complexity of multiplexing the pattern compared to the case where only valid ones are retained. Since the SC decoder is sequential, the special node vectors $V$ (Fig. 4c) are read from the register one at a time through a $\frac{N}{M} \cdot (M-1)$ to-1 MUX. A $\log_2 N$-bit counter is used to generate the register-based memory addresses. At each special node iteration search, the next pattern to read from the registers array depends on the size and address of the previous one. The complexity of the proposed architecture increases linearly with code length $N$. But since the special nodes are searched only upon request, a low complexity serial implementation is favoured. To do this, a $N$-to-$M$ multiplexer is used to process serially the frozen set bits in groups of $M$. The architecture of a serial search of special nodes is depicted in Fig. 4c.

## 4 Synthesis results and comparisons

As a proof of concept, a decoder architecture fully compliant with the 5G NR polar code has been designed. The proposed architecture is generic with respect to the parameters $P$, $L$ and $M$. In addition to the proposed special node list decoding and special node identification unit, the architecture includes: $L \times P$ *Processing Elements (PE)* to compute and update the LLRs, *Multi-bits Partial Sum Network* to produce PS, *CRC bits calculation unit* to perform CRC check operations in the case of multi-bit decoding and which operates in parallel to the decoding process. The *Control unit* integrates a finite state machine designed to generate all the control signals needed for the different decoder components. In order to manage candidates competition, crossbars were also designed whether for copying data to some freed memory locations or to ensure a correct routing of data as a result of the use of pointer memories.

The devised architecture has been described in VHDL. The FPGA used for synthesis was a Xilinx Virtex 7-xc7vx485t device. The number of bits used to represent internal LLR, PM and channel LLR values is $Q_i = 6$, $Q_p = 7$ and $Q_c = 4$, respectively. Eight PEs are instantiated in the architecture for each of the $L$ paths, where $L = 8$. The maximum size of special nodes $M$ is set to 32. Internal LLRs and partial sum bits are stored in the FPGA dual-port Block RAMs (BRAMs) while input LLRs and partial sum memory addresses are stored using look-up tables (LUTs). Decoding latency is the main critical performance metric when considering the 5G NR polar codes that protect the control channel. A target end-to-end latency of 0.5 ms implies a physical layer latency of 50 $\mu$s [6, 13]. Flexibility and low latency are the driving priority for the design of this channel decoder, which is the main demanding component of the physical layer.

*Table 1: Average and maximum latency measured by the proposed decoder.*

| | | | Worst-case latency | | Average Latency | |
|---|---|---|---|---|---|---|
| Physical control channel | N | # code | cc | [$\mu$s] | cc | [$\mu$s] |
| Downlink | 64 | 435 | 146 | 1.35 | 121 | 1.11 |
| | 128 | 3451 | 284 | 2.63 | 207 | 1.91 |
| | 256 | 9452 | 483 | 4.47 | 376 | 3.47 |
| | 512 | 2286 | 780 | **7.22** | 719 | 6.65 |
| Uplink | 1024 | 3491 | 2583 | **23.91** | 1914 | 17.72 |

The latency for decoding one codeword is not constant and is highly affected by the choice of the operating code length and code rate. In order to give a full analysis of the latency, all combinations of code lengths and code rates ranging from $R = 1/8$ to $R = 5/6$ are evaluated through simulations. Table 1 provides the average and the worst-case latency for decoding one codeword. The latency is measured in number of clock cycles (cc) and in $\mu$s considering the maximum clock frequency $f_{\text{max}} = 108$ MHz. The results show that the maximum latency recorded by the decoder is 23.91 $\mu$s which is 2.1 times lower than the physical layer latency constraint. This worst-case latency appears in the uplink scenario, whereas it is only 7.22 $\mu$s in the downlink.

Logic synthesis and performance results are summarized in Table 2. Implementations that are fully compatible with 5G code specifications, in terms of code structure and flexibility, allow for direct and fair comparisons. Therefore, the recently available Xilinx polar decoder [20] is the most relevant reference with respect to the available decoder designs in the literature. Performance results of Xilinx decoder are available, yet with no published details on the architecture. Compared to this decoder, our proposed architecture has 60% and 70% less decoding latency for the uplink (1024,512) and the downlink (512,40) polar codes, respectively. Our design consumes 38% less Flip-Flops (FFs), but 11% more LUTs and 25% more BRAMs. The throughput, however, does not compare favorably, yet still compliant with the 5G NR requirement for this code used for the control channel.

In order to extend the comparison, we have considered recent designs that targeted FPGA implementation with similar code length, yet not compliant with 5G NR polar codes. For that, a second configuration of our proposed architecture has been designed and synthesized with $L = 4$ while keeping the number of PEs per list unchanged. Compared to the folding polar decoder of [12], our decoder uses 74% less LUTs and 6% less FFs while decoding the (1024,512) code in less than half the time (converted in clock cycles). However, without any reported clock frequency, the latency comparison is not complete. Compared to the stochastic SCL decoder of [11] targeting wearable and IoT devices with strict hardware constraints, our decoder supporting rate and frame size flexibility exhibits 109 times less latency while requiring 3.2 times and 4.4 times the numbers of LUTs and FFs, respectively.

*Table 2: Comparison with several FPGA-based SCL Architectures.*

| Decoder | [12] | [11] | **This work** | Xilinx [20] | | **This work** | |
|---|---|---|---|---|---|---|---|
| List size | | 4 | | | 8 | | |
| Algorithm | SCL | SSCL[1] | SSCL[2] | NA | | SSCL[2] | |
| Flexibility | Limited | Limited | High | High | | High | |
| Quantization | NA | NA | (4,6,7) | 8 | | (4,6,7) | |
| # PE | 64 | 64 | 8 | NA | | 8 | |
| $f_{\max}$ (MHz) | N/A | 445.2 | 108 | 223 | | 108 | |
| FPGA Device | Stratix V | Stratix V | xc7vx485t | xc7vx485t | | xc7vx485t | |
| ALMs/LUTs | 101160 | 8146 | **26049** | 45569 | | **51262** | |
| FFs | 13544 | 2862 | **12603** | 33063 | | **20270** | |
| BRAMs | 0 | 0 | **34** | 51.5 | | **68** | |
| (N,K) | (1024,512) | (1024,512) | (1024,512) | (512,40) | (1024,512) | (512,40) | (1024,512) |
| Latency ($\mu$s) | 4064[3] | 1177 | **10.73** | 11.35 | 46.41 | **4.76** | **18.37** |
| TP (Mbps) | N/A | 0.452 | 47.7 | 28.68 | 88.4 | 8.4 | 34.4 |

[1] Stochastic SCL with 2-level decoding
[2] Simplified SCL with four constituent codes
[3] Number of clock cycles

## 5 Conclusion

In this paper we proposed an original hardware architecture for decoding the 5G NR polar codes of the uplink and the downlink control information channel. Thanks to a special node identifier, the proposed decoder continues to benefit from tree pruning techniques to speed-up the decoding whilst maintaining compliance with the 5G NR and the various defined combinations of code rate and code length. Hence, measured throughput and latency values of the proposed decoder obtained with FPGA target are able to meet 5G requirements. Moreover, synthesis results have shown a hardware efficiency that compares favourably with state-of-the-art FPGA implementations of polar decoders. Finally, the designed architecture reduces the decoding latency compared to the recently available Xilinx 5G polar decoder.

## References

1. 3GPP: TS NR Multiplexing and Channel Coding. Release 15 V15.6.0 TS, 38.212 (2019), `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214`
2. Alamdar-Yazdi, A., Kschischang, F.R.: A simplified successive-cancellation decoder for polar codes. IEEE communications letters **15**(12), 1378–1380 (2011)
3. Arikan, E.: Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Transactions on information Theory **55**(7), 3051–3073 (2009)
4. Bian, X., Dai, J., Niu, K., He, Z.: A low-latency SC polar decoder based on the sequential logic optimization. In: International Symposium on Wireless Comm. Systems (ISWCS) (2018)

5. Egilmez, Z.B.K., Xiang, L., Maunder, R.G., Hanzo, L.: The development, operation and performance of the 5g polar codes. IEEE Communications Surveys & Tutorials **22**(1), 96–122 (2019)
6. Fettweis, G.P.: The Tactile Internet: Applications and Challenges. IEEE Vehicular Technology Magazine **9**(1), 64–70 (2014)
7. Hashemi, S.A., Condo, C., Gross, W.J.: A fast polar code list decoder architecture based on sphere decoding. IEEE Transactions on Circuits and Systems I: Regular Papers **63**(12), 2368–2380 (2016)
8. Hashemi, S.A., Condo, C., Gross, W.J.: Simplified successive-cancellation list decoding of polar codes. In: IEEE International Symposium on Information Theory (ISIT). pp. 815–819 (2016). https://doi.org/10.1109/ISIT.2016.7541412
9. Hashemi, S.A., Condo, C., Gross, W.J.: Fast simplified successive-cancellation list decoding of polar codes. In: IEEE Wireless Communications and Networking Conference Workshops (WCNCW). pp. 1–6. IEEE (2017)
10. Leroux, C., Tal, I., Vardy, A., Gross, W.J.: Hardware architectures for successive cancellation decoding of polar codes. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1665–1668. IEEE (2011)
11. Liang, X., Wang, H., Shen, Y., Zhang, Z., You, X., Zhang, C.: Efficient stochastic successive cancellation list decoder for polar codes. Science China Information Sciences **63**(10), 1–19 (2020)
12. Liang, X., Yang, J., Zhang, C., Song, W., You, X.: Hardware efficient and low-latency ca-scl decoder based on distributed sorting. In: 2016 IEEE Global Communications Conference (GLOBECOM). pp. 1–6. IEEE (2016)
13. Maunder, R.G.: The 5G channel code contenders. ACCELERCOMM white paper pp. 1–13 (2016)
14. Niu, K., Chen, K.: Crc-aided decoding of polar codes. IEEE communications letters **16**(10), 1668–1671 (2012)
15. Roy, S.J., Lakshminarayanan, G., Ko, S.B.: High speed architecture for successive cancellation decoder with split-g node block. IEEE Embedded Sys. Letters (2020)
16. Sarkis, G., Giard, P., Vardy, A., Thibeault, C., Gross, W.J.: Fast polar decoders: Algorithm and implementation. IEEE Journal on Selected Areas in Communications **32**(5), 946–957 (2014). https://doi.org/10.1109/JSAC.2014.140514
17. Sarkis, G., Giard, P., Vardy, A., Thibeault, C., Gross, W.J.: Fast list decoders for polar codes. IEEE Journal on Selected Areas in Comm. **34**(2), 318–328 (2016)
18. Shrestha, R., Sahoo, A.: High-speed and hardware-efficient successive cancellation polar-decoder. IEEE Trans. on Circuits and Systems II **66**(7), 1144–1148 (2018)
19. Tal, I., Vardy, A.: List decoding of polar codes. IEEE Transactions on Information Theory **61**(5), 2213–2226 (2015)
20. Xilinx: IP Polar Encoder/Decoder. `https://www.xilinx.com/products/intellectual-property/ef-di-polar-enc-dec.html#overview` (2021)
21. Yuan, B., Parhi, K.K.: Low-latency successive-cancellation polar decoder architectures using 2-bit decoding. IEEE Transactions on Circuits and Systems I: Regular Papers **61**(4), 1241–1254 (2013)
22. Zhang, C., Parhi, K.K.: Low-latency sequential and overlapped architectures for successive cancellation polar decoder. IEEE Transactions on Signal Processing **61**(10), 2429–2441 (2013)