# Memory-Efficient Polar Decoders

Seyyed Ali Hashemi, *Student Member, IEEE*, Carlo Condo, *Member, IEEE*, Furkan Ercan, *Student Member, IEEE*, and Warren J. Gross, *Senior Member, IEEE*

***Abstract*— Polar codes have gained a great amount of attention in the past few years, since they can provably achieve the capacity of a symmetric channel with a low-complexity encoding and decoding algorithm. As a result, polar codes have been selected as a coding scheme in the 5th generation wireless communication standard. Among different decoding schemes, successive-cancellation (SC) and SC list decoding yield good trade-off between error-correction performance and hardware implementation cost. However, both families of algorithms have large memory requirements. In this paper, we propose a set of novel techniques that aim at reducing the high-memory cost of SC-based decoders. These techniques are orthogonal to the specific decoder architecture considered, and can be applied on top of existing memory reduction techniques. We have designed and implemented different polar decoders on FPGA and also synthesized them in 65 nm TSMC CMOS technology to verify the effectiveness of the proposed memory reduction techniques. The benchmark decoders yield comparable or lower area occupation than the state of the art: the results show that the proposed methods can save up to 46% memory area occupation and 42% total area occupation compared with benchmark SC-based decoders.**

***Index Terms*— Polar codes, successive-cancellation decoding, list decoding, hardware implementation.**

## I. Introduction

DUE to their inherent capacity-achieving property and low-complexity encoding and decoding [1], polar codes have gained great amount of attention in the past few years, to the extent that they have been selected for the 5th generation (5G) wireless communication standard [2]. Successive-cancellation (SC) decoder was the first decoder used to decode polar codes and it was shown that as the length of the code tends to infinity, polar codes under SC decoding reach the capacity of a symmetric channel [1]. However, at finite practical code lengths, SC falls short of providing a reasonable error-correction performance. In order to address this issue, SC list (SCL) decoding closed the gap between the error-correction performance of an SC decoder and a maximum likelihood (ML) decoder while keeping the overall decoder complexity at a moderate level. In fact, SCL employs a list of SC decoders working in parallel and at every bit-estimation, a list of candidate codewords is allowed to survive. The decoder then selects the most reliable codeword as the output. Since the error-correction performance of polar codes under ML decoding is mediocre, a cyclic redundancy check (CRC) is concatenated to polar codes to help find the correct candidate among the list of codewords. The CRC-aided SCL decoder (SCL-CRC) improved the error-correction performance of polar codes such that they could outperform state-of-the-art low-density parity-check (LDPC) codes [3]. An alternative to SC-based decoders are belief propagation decoders. Belief propagation decoders require a higher implementation complexity with respect to SC, and many iterations are necessary to match the error-correction performance of SC decoders [4]. Efforts to reduce the complexity and improve the latency of such decoders have been recently made in [5]–[8].

Although 5G technology is not standardized yet, it is expected to require tens to thousands times better performance parameters than the 4th generation (4G) technology. Among these parameters are the speed and the area occupation in the hardware implementation of devices. Channel decoders, such as SC and SCL (SCL-CRC), are thus required to have high throughput with low area occupation. These requirements are challenging for SC and SCL decoders: the serial nature of the algorithms rises latencies and limits throughput [9], [10], while high memory usage increases their area occupation [10]. Solutions have been proposed to increase the speed of both SC and SCL, resulting in fast simplified SC (Fast-SSC) [11] and simplified SCL (SSCL) [12] decoding algorithms.

The reduction of memory requirements of SCL decoders, that are higher than those of SC, has been addressed in [13]–[15]. However, in [13], the design need to be re-evaluated when the code changes. The solution presented in [14] is based on log-likelihood (LL) messages, which require more memory than its LL ratio (LLR) counterparts, and the sphere-decoding-based technique in [15] suffers from error-correction performance degradation as the code length increases. Effective memory-reduction techniques that do not incur performance loss are needed, especially within the challenging 5G framework.

This paper is an extension of the work in [16], where the concept of partitioned SCL (PSCL) was first introduced. Here, we propose a CRC selection scheme which improves the error-correction performance of PSCL. We also propose a set of memory reduction techniques for SC-based decoders, that are orthogonal to the decoder architecture. In particular, aside from partitioning, we present a memory sharing technique that does not introduce any approximation and is independent of the decoder hardware structure, a study on quantization that allows to reduce the bits necessary to represent the channel LLR values, and a memory optimization method for SRAM-based SC and Fast-SSC decoders. Decoder architectures implementing the proposed techniques are designed and synthesized

Manuscript received February 16, 2017; revised May 27, 2017 and August 21, 2017; accepted October 13, 2017. Date of publication October 18, 2017; date of current version December 14, 2017. This paper was recommended by Guest Editor Farhana Sheikh. *(Corresponding author: Seyyed Ali Hashemi.)*

The authors are with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 0G4, Canada (e-mail: seyyed.hashemi@mail.mcgill.ca; carlo.condo@mail.mcgill.ca; furkan.ercan@mail.mcgill.ca; warren.gross@mcgill.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2017.2764421

in 65 nm TSMC CMOS technology, showing up to 46% memory area and 42% total area occupation reduction, with no impact on the decoding speed and negligible error-correction performance degradation. In Section II, we introduce polar codes and summarize the most common decoding algorithms. Section III details the proposed memory reduction techniques. Section IV details the modifications necessary to implement the memory reduction techniques on state-of-the-art polar decoders. Implementation results are given in Section V, and conclusions are drawn in Section VI.

## II. Polar Codes

$\mathcal{P}(N, K)$ represents a polar code of length $N$ with $K$ information bits. Therefore, the rate of a polar code can be represented as $R \triangleq K/N$. Polar codes can be constructed by the concatenation of two polar codes of length $N/2$. This recursive construction can be represented as the matrix product of the input vector $\mathbf{u} = \{u_0, u_1, \ldots, u_{N-1}\}$ and the generator matrix $\mathbf{G}_N$ as

$$\mathbf{x} = \mathbf{u}\mathbf{G}_N, \tag{1}$$

where $\mathbf{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ is the sequence of coded bits, $n = \log_2 N$, and $\mathbf{G}_N = \mathbf{B}_N\mathbf{G}^{\otimes n}$, where $\mathbf{B}_N$ is a bit-reversal permutation matrix, and $\mathbf{G}^{\otimes n}$ is the $n$-th Kronecker product of the polarizing matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2}$$

Determining the bit positions in $\mathbf{u}$ that can carry the information bits requires splitting the $N$ polarizing bit-channels in two groups. It was shown that as $N$ approaches infinity, the bit-channels tend to become either noiseless or pure noise [1] and that the proportion of noiseless bit-channels equals the capacity of the channel. For finite $N$, $K$ reliable bit-channels are selected [17] and the information bits are assigned to them. The remaining bit-channels are set to a predefined value (usually 0) known at the decoder and thus they are called frozen bits with set $\mathcal{F}$. The resulting $\mathbf{u}$ is coded into $\mathbf{x}$ using (1). $\mathbf{x}$ is then modulated and transmitted through the channel. We consider the additive white Gaussian noise (AWGN) channel and binary phase-shift keying (BPSK) modulation in this paper.

### A. Successive-Cancellation Decoding

SC decoding can be represented on a binary tree as shown in Fig. 1. The received vector from the channel is fed to the decoder at stage $n$. This vector can be represented by channel LLR values in accordance with $\boldsymbol{\alpha}_n^{0\to N-1} = \{\alpha_n^0, \alpha_n^1, \ldots, \alpha_n^{N-1}\}$. At each stage $s$ of SC decoding, the elements of the vector of internal LLR values $\boldsymbol{\alpha}_s^{0\to N-1} = \{\alpha_s^0, \alpha_s^1, \ldots, \alpha_s^{N-1}\}$ which is composed of $N/2^s$ vectors of length $N_s = 2^s$ bits $\boldsymbol{\alpha}_s^{iN_s\to\,(i+1)N_s-1} = \{\alpha_s^{iN_s}, \alpha_s^{iN_s+1}, \ldots, \alpha_s^{(i+1)N_s-1}\}$ are calculated as

$$\alpha_s^i = \operatorname{sgn}(\alpha_{s+1}^i)\operatorname{sgn}(\alpha_{s+1}^{i+N_s})\min(|\alpha_{s+1}^i|, |\alpha_{s+1}^{i+N_s}|), \tag{3}$$

$$\alpha_s^{i+N_s} = \alpha_{s+1}^{i+N_s} + (1-2\beta_s^i)\alpha_{s+1}^i, \tag{4}$$

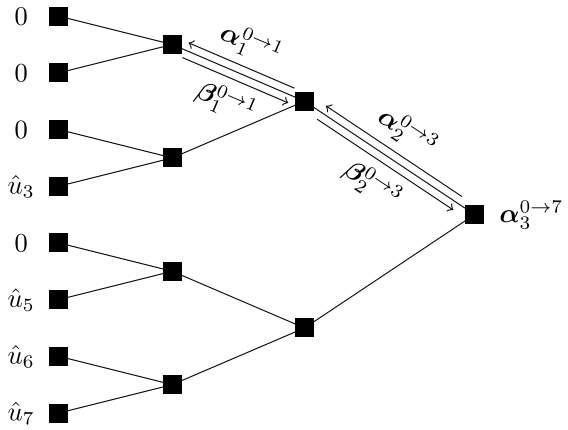


Fig. 1. Polar code SC decoding tree for $\mathcal{P}(8, 4)$ when $\{u_0, u_1, u_2, u_4\} \in \mathcal{F}$.

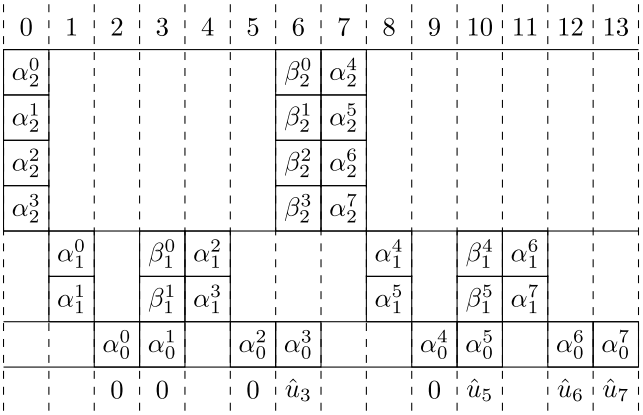


Fig. 2. SC scheduling for $\mathcal{P}(8, 4)$.

where (3) is a hardware-friendly formulation proposed in [9], that causes negligible error-correction performance degradation. The hard-decision estimates $\boldsymbol{\beta}_s$ are calculated as

$$\beta_s^i = \beta_{s-1}^i \oplus \beta_{s-1}^{i+N_s}, \tag{5}$$

$$\beta_s^{i+N_s} = \beta_{s-1}^{i+N_s}, \tag{6}$$

where $\oplus$ is the bitwise XOR operation. At a leaf node, the $i$-th bit $\hat{u}_i$ is estimated as

$$\hat{u}_i = \beta_0^i = \begin{cases} 0, & \text{if } i \in \mathcal{F} \text{ or } \alpha_0^i \geq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

In SC decoding, each bit estimation is dependent on the value of all the previous bits. Thus, the conventional SC decoder can complete the decoding process in $2N - 2$ time-steps, with the scheduling depicted in Fig. 2. It can be seen that three sets of memory are required to decode polar codes using SC: the channel LLR memory, the internal LLR memory, and the $\beta$ memory. Let us consider the channel LLR values are quantized with $Q_{\alpha_C}$ bits and the internal LLR values are quantized with $Q_{\alpha_I}$ bits. Since each $\beta$ value is represented with one bit, the total memory requirements for a SC decoder can be calculated as

$$M_{\text{SC}} = N Q_{\alpha_C} + (N-1)\, Q_{\alpha_I} + N - 1. \tag{8}$$

### B. Successive-Cancellation List Decoding

SCL decoding improves the error-correction performance of SC decoding by employing $L$ SC decoders working in parallel. Each information bit is estimated as either 0 or 1, thus doubling the number of considered candidate codewords at each bit-estimation step $i$. To limit the exponential growth in the complexity of SCL decoder, only $L$ paths are allowed to survive among the $2L$ created by the decoder. A path metric (PM) is used to determine which of the $L$ candidates can survive as [10]

$$\mathrm{PM}_{-1_l} = 0,$$
$$\mathrm{PM}_{i_l} = \begin{cases} \mathrm{PM}_{i-1_l} + |\alpha^i_{0_l}|, & \text{if } \hat{u}_{i_l} \neq \dfrac{1-\mathrm{sgn}\left(\alpha^i_{0_l}\right)}{2}, \\ \mathrm{PM}_{i-1_l}, & \text{otherwise}, \end{cases} \quad (9)$$

where $l$ is the path index.

At the end of the decoding process, the SCL decoder selects the final candidate out of the $L$ surviving ones, as the one with the best PM. The SCL error-correction performance can only be as good as the ML decoder. If a CRC is concatenated to polar code, it can help select the correct codeword among the final $L$ candidates. It should be noted that the CRC is only used at the end of the SCL-CRC decoding process. In addition, care needs to be taken for the selection of CRC length $C$ to provide a good error-correction performance.

The memory requirements of SCL (SCL-CRC) decoding are higher than that of the SC decoding since $L$ paths of the candidate codewords need to be stored. In addition, the PM for each path also needs to be stored. Let us consider $Q_{\mathrm{PM}}$ bits are used to store the PMs. The memory requirement of a SCL decoder can be written as

$$M_{\mathrm{SCL}} = N Q_{\alpha_C} + L\,(N-1)\,Q_{\alpha_I} + L Q_{\mathrm{PM}} + L\,(2N-1). \quad (10)$$

It should be noted that in SCL decoding, $LN$ bits need to be stored for the final codeword candidates as opposed to SC decoding.

## III. Memory Reduction Techniques

In this section, we propose new methods to reduce the memory requirements of SC and SCL decoders. These techniques are orthogonal to the decoder architecture.

### A. Partitioned SCL Decoding

PSCL decoding allows to significantly reduce the memory requirements associated with SCL decoders. The idea is to break the decoding tree into two parts, i.e. the top and the bottom of the tree: the latter is composed of $P$ sub-trees (partitions), and each partition is decoded with a SCL-CRC decoder. Instead of using the CRC to find the correct codeword at the end of SCL-CRC decoding process, PSCL uses the CRC to find a single correct codeword for each partition. It then passes this candidate to the top of the tree, where standard SC decoding is performed, and until the next partition is encountered. Therefore, it is not necessary to store $L$ full trees as in SCL, but only $L$ copies of the part of the tree

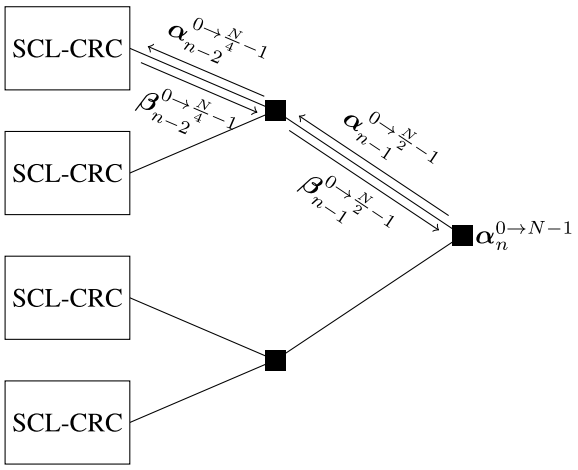


Fig. 3. PSCL tree structure for $P = 4$.

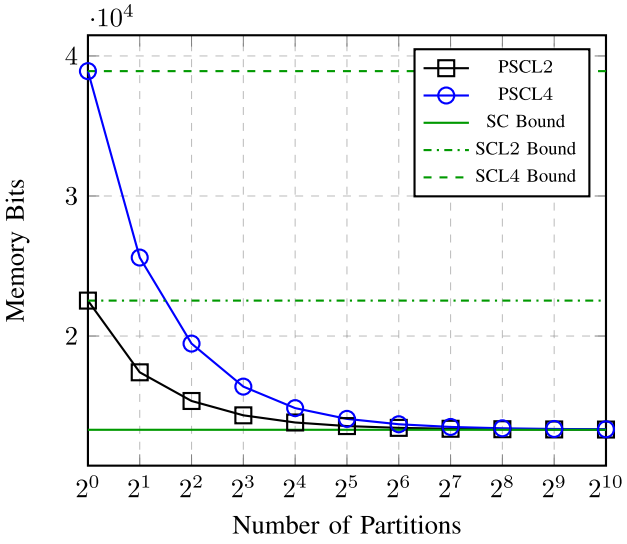


Fig. 4. PSCL memory requirements for a polar code of length 1024 when $L = \{2, 4\}$. PSCL$L$ and SCL$L$ represent PSCL and SCL decoding with list size $L$.

contained in the partitions. Moreover, memory can be shared among the $P$ partitions, which results in significant savings as $P$ increases. Fig. 3 shows the PSCL decoding tree when $P = 4$. As described in [1] and in Section II, a polar code of length $N$ can be seen as the concatenation of two polar codes of length $N/2$: thus, each partition is a polar code of length $N/P$.

The memory requirement of PSCL is bound between those of SC ($P = N$) and SCL ($P = 1$) decoders as

$$M_{\mathrm{PSCL}} = N Q_{\alpha_C} + \left(\sum_{k=1}^{\log_2 P} \frac{N}{2^k} + L\left(\frac{N}{P} - 1\right)\right) Q_{\alpha_I} + L Q_{\mathrm{PM}} + \sum_{k=1}^{\log_2 P} \frac{N}{2^k} + L\left(\frac{2N}{P} - 1\right), \quad (11)$$

where $2 \leq P < N$. It is worth noting that as the number of partitions increases, the memory usage decreases exponentially toward the SC bound as depicted in Fig. 4 for a code of

length 1024. However, in the conventional PSCL algorithm, this memory saving is obtained at the cost of error-correction performance degradation.

There are two main reasons associated with this performance degradation. First, PSCL uses SC decoding at the top of the polar code tree which can cause error-correction performance degradation with respect to SCL. Second, the uniform distribution of CRCs between partitions in the conventional PSCL may result in significant deterioration in error-correction performance. It was shown in [18] that the first issue can be mitigated by carefully constructing polar codes for a different channel. Moreover, a successive CRC assignment (SCA) scheme was developed to identify the optimal CRC length for each partition. However, there are two main issues associated with the SCA approach. The first issue is that the identification of the optimal CRC length for each partition is performed by finding the error-correction performance of each partition in a serial manner which requires the knowledge of the correct codeword for previously decoded partitions. The second issue stems from the fact that SCA finds the optimal CRC lengths of partitions without having any constraint on them. This is in contrast with the conventional PSCL approach, in which in order to keep the effective rate of polar codes constant, the sum of CRC lengths for partitions is kept at $C$.

In order to tackle the above issues, we use Gaussian approximation (GA) to find the error-correction performance of each partition in parallel, without any recourse to the information from other partitions. After finding the error-correction performance of each partition for a given $E_b/N_0$ and CRC length, we impose a constraint on the sum of CRC lengths of partitions to find the optimal value of CRC length for each partition such that the effective rate of the code remains unchanged at $\frac{K+C}{N}$ for a fair comparison with SCL-CRC decoding with CRC length $C$. We further show that this constraint can be modified to only select the CRC lengths from a set of practical ones.

GA was first used to reduce the complexity of polar code construction for AWGN channel in [19]. In this setting, the channel LLR values, $\alpha_n^i$, have a normal distribution $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$, where $\sigma$ represents the standard deviation of the AWGN channel, and the transmission of the all-zero codeword is considered. The intermediate LLR values at stage $s$ of the SC decoding tree can be approximated as having a normal distribution $\mathcal{N}(\mu_s^i, 2\mu_s^i)$. The value of $\mu_s^i$ can be calculated recursively as

$$\mu_s^i = \phi^{-1}\left(1-\left(1-\phi\left(\mu_{s+1}^i\right)\right)^2\right), \tag{12}$$

$$\mu_s^{i+N_s} = 2\mu_{s+1}^{i+N_s}, \tag{13}$$

where

$$\phi(\mu) = \begin{cases} 1-\frac{1}{\sqrt{4\pi\mu}}\int_{-\infty}^{\infty}\tanh\frac{u}{2}\mathrm{e}^{-\frac{(u-\mu)^2}{4\mu}}\,du, & \mu>0, \\ 1, & \mu=0. \end{cases} \tag{14}$$

It should be noted that in (12) and (13), $\mu_{s+1}^i = \mu_{s+1}^{i+N_s}$. GA determines the channel seen by each partition, provided that all the sub-channels are approximated as AWGN channels. We can therefore obtain the error-correction performance of each partition independently. Although the frame error rate (FER) performance of polar codes under SC decoding can be calculated with GA [19], to the best of our knowledge, there is no analytical formula for finding the FER performance of polar codes under SCL or SCL-CRC decoding. Therefore, we use a simulation-based approach to find the FER of each partition under SCL-CRC decoding. It is worth mentioning that once the code parameters change, the FER has to be recalculated for each partition. However, this approach is performed only once and is done off-line so there is no overhead in the hardware implementation complexity. GA approximates the intermediate LLR values calculated by SC decoding by assuming that the previous bits are decoded correctly. Therefore, the FER of polar codes under PSCL decoding can be calculated as the sum of the FER performance of the partitions simulated with GA-obtained AWGN channels.

The GA-obtained channels allow to find the optimal CRC length for each partition. Let us consider $\mathbf{c} = \{c_0, c_1, \ldots, c_{P-1}\}$, representing the set of CRC lengths for the PSCL partitions. In the conventional PSCL algorithm, the CRC length for partition $j$ is selected as

$$c_j = \frac{C}{P} \text{ bits}. \tag{15}$$

While simple, this method can introduce significant error-correction performance loss as the number of partitions increases [16]. Let us consider an AWGN channel with a certain $E_b/N_0$. We can calculate the standard deviation of this channel as

$$\sigma = \sqrt{\frac{10^{-\frac{E_b/N_0}{10}}}{2R}}. \tag{16}$$

Therefore,

$$\mu_n^i = 4R10^{\frac{E_b/N_0}{10}}. \tag{17}$$

We can now use (12) and (13) to determine the channels that are seen by the two partitions in stage $n-1$. The $E_b/N_0$ values of the channels seen by each partition can be calculated as

$$E_b/N_{0_0} = -10\log_{10}\left(\frac{4R_0}{\mu_{n-1}^i}\right), \tag{18}$$

$$E_b/N_{0_1} = -10\log_{10}\left(\frac{4R_1}{\mu_{n-1}^{i+N/2}}\right), \tag{19}$$

where $E_b/N_{0_j}$ represents the $E_b/N_0$ value of the channel seen by partition $j$, and $R_j$ is the rate of the polar code in partition $j$. A recursive application of (16)-(19) results in the channels seen by partitions at a lower stage.

As an example, let us assume PSCL decoding of $\mathcal{P}(1024, 512)$ with an AWGN channel with $E_b/N_0 = 3$ dB, while the code is optimized for $E_b/N_0 = 2$ dB. Table I shows the rate of the polar codes in the various partitions, and Table II summarizes the $E_b/N_0$ values of the channels seen by the partitions.

It is now possible to find the FER of each partition based on the $E_b/N_0$ values of Table II while considering different values of CRC length for each partition. Fig. 5 and Fig. 6 show

TABLE I

RATE OF POLAR CODES SEEN IN PSCL DECODING OF $\mathcal{P}(1024, 512)$ FOR $P = 2$ AND $P = 4$

| $P$ | Rate | | | |
|---|---|---|---|---|
| 1 | 512/1024 | | | |
| 2 | 135/512 | | 377/512 | |
| 4 | 19/256 | 116/256 | 141/256 | 236/256 |

TABLE II

$E_b/N_0$ VALUES OF THE CHANNELS SEEN IN PSCL DECODING OF $\mathcal{P}(1024, 512)$ FOR $P = 2$ AND $P = 4$

| $P$ | $E_b/N_0$ [dB] | | | |
|---|---|---|---|---|
| 1 | 3 | | | |
| 2 | 3.3217 | | 4.3293 | |
| 4 | 5.2449 | 3.9805 | 4.1832 | 6.3636 |

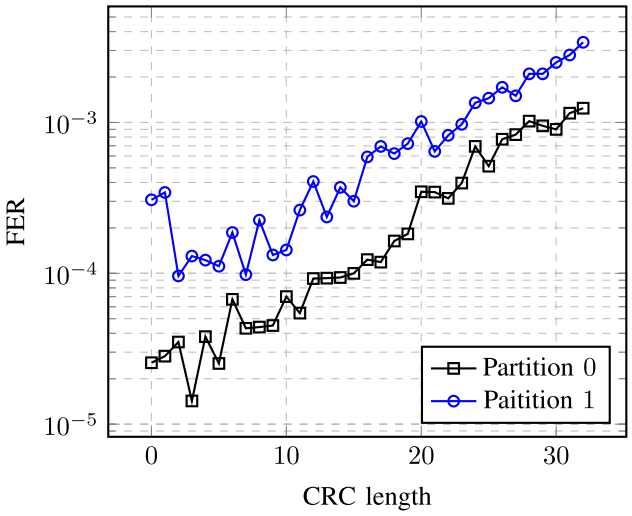


Fig. 5. Effect of CRC length on the FER of each partition for $\mathcal{P}(1024, 512)$ when $P = 2$ and $L = 2$. The FER of the two partitions were derived independently using GA for $E_b/N_0 = 3$ dB.

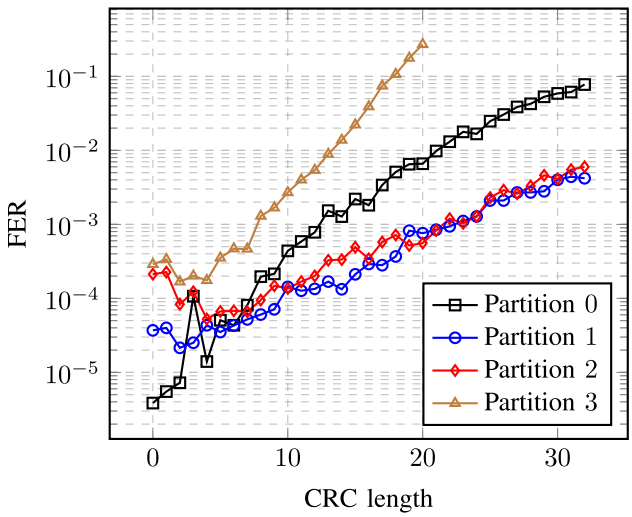


Fig. 6. Effect of CRC length on the FER of each partition for $\mathcal{P}(1024, 512)$ when $P = 4$ and $L = 2$. The FER of the four partitions were derived independently using GA for $E_b/N_0 = 3$ dB.

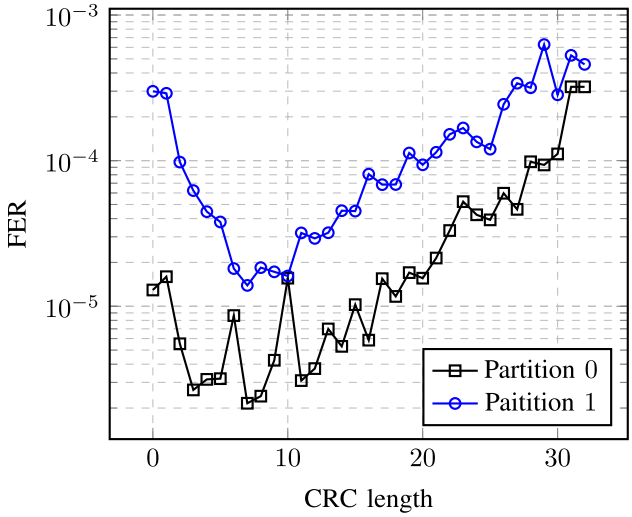


Fig. 7. Effect of CRC length on the FER of each partition for $\mathcal{P}(1024, 512)$ when $P = 2$ and $L = 4$. The FER of the two partitions were derived independently using GA for $E_b/N_0 = 3$ dB.

the effect of CRC length on the FER of partitions when the channel seen by $\mathcal{P}(1024, 512)$ has $E_b/N_0 = 3$ dB for $P = 2$, and $P = 4$, respectively. In these figures, SCL decoding with $L = 2$ was used to derive the FER of each partition. Fig. 7 and Fig. 8 are plotted similar to Fig. 5 and Fig. 6, but they use SCL with $L = 4$ instead. It can be seen that there is a specific $c_j$ which leads to an optimal error-correction performance for partition $j$ for every $L$ and when $E_b/N_0 = 3$ dB. However, we have the constraint that

$$\sum_{j=0}^{P-1} c_j = C. \tag{20}$$

Therefore, we have to solve the following optimization problem

$$\underset{\sum_{j=0}^{P-1} c_j = C}{\arg\min} \sum_{j=0}^{P-1} \mathrm{FER}_j(c_j), \tag{21}$$

where $\mathrm{FER}_j(c_j)$ is the FER of partition $j$ when the CRC length is $c_j$. The process for selecting a good CRC length for partitions is summarized in Algorithm 1.

**Algorithm 1** Determining CRC Length for Each Partition.

**Input**: $L, P, C, E_b/N_0$
**Output**: **c**
Find $E_b/N_0$ of the $P$ partitions using (16)-(19)
**for** $j \leftarrow 0$ ***to*** $P - 1$ **do**
  **for** $c_j \leftarrow 0$ ***to*** $C$ **do**
    Find $\mathrm{FER}_j(c_j)$ for given $L$
  **end**
**end**
Solve (21)
**Result**: $c_j$ for $0 \leq j < P$.

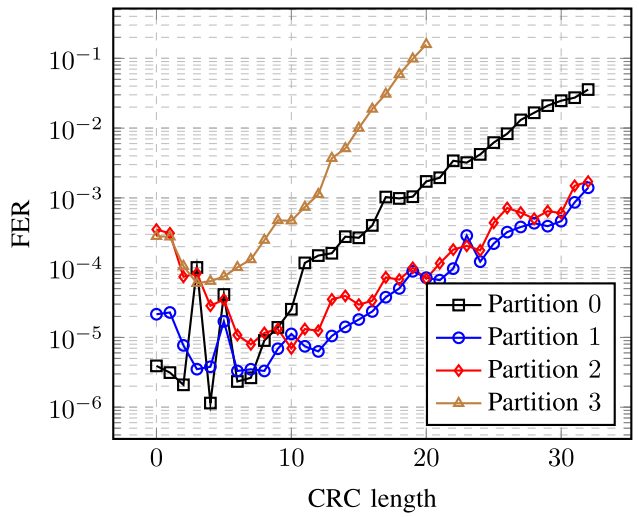


Fig. 8. Effect of CRC length on the FER of each partition for $\mathcal{P}(1024, 512)$ when $P = 4$ and $L = 4$. The FER of the four partitions were derived independently using GA for $E_b/N_0 = 3$ dB.

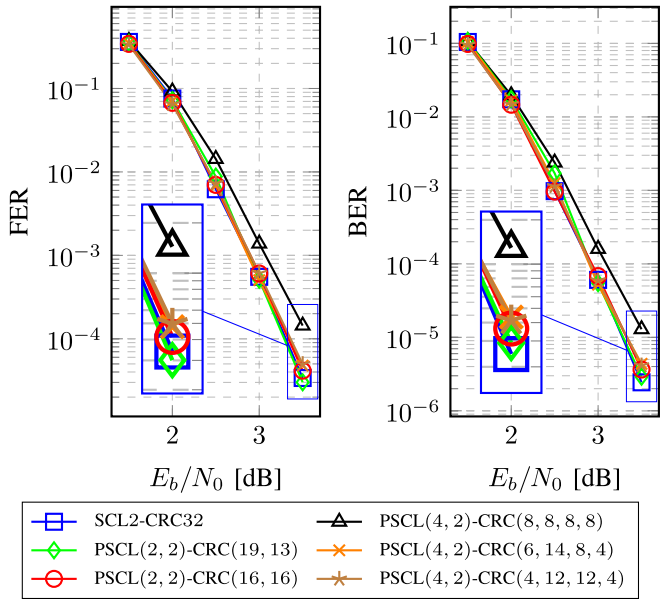


Fig. 9. FER and BER performance for PSCL decoding of $\mathcal{P}(1024, 512)$ when $L = 2$. The code is optimized for $E_b/N_0 = 2$ dB.

Algorithm 1 causes the CRC lengths of each partition to be any value between 0 and $C$. In practical applications, we are usually constrained to have a set of standard CRC lengths [20]. Let us consider we are constrained to have CRC lengths which belong to the set $\mathcal{C}$ whose elements can be selected based on a specific application. Therefore, (21) can be rewritten as

$$\underset{\substack{\sum_{j=0}^{P-1} c_j = C \\ c_j \in \mathcal{C}}}{\arg\min} \sum_{j=0}^{P-1} \mathrm{FER}_j(c_j). \tag{22}$$

In this paper, we solve (21) and (22) with an exhaustive search on the sum of FER values of partitions for different CRC lengths and we consider $\mathcal{C} = \{4, 8, 12, 16, 20\}$.

Fig. 9 and Fig. 10 show the FER and bit error rate (BER) curves for SCL-CRC and PSCL with $L = 2, 4$ and $P = 2, 4$. The SCL-CRC decoders are labelled as SCL$L$-CRC$C$, which represents SCL-CRC decoding with list size $L$ and using a CRC of size $C$. The PSCL decoders are labelled as PSCL($P$,$L$)-CRC($c_0$,$c_1$,...,$c_{P-1}$), where $c_j$ is the length of the CRC assigned to partition $j$. They compare the error-correction performance obtained with CRC lengths calculation of Algorithm 1, the additional constraint of (22), and the conventional CRC lengths calculation of (15). It should be noted that for $L = 2$, Algorithm 1 results in PSCL(2,2)-CRC(19,13) and PSCL(4,2)-CRC(6,14,8,4), while adding the constraint in (22) results in PSCL(2,2)-CRC(16,16) and PSCL(4,2)-CRC(4,12,12,4). For $L = 4$, Algorithm 1 results in PSCL(2,4)-CRC(20,12) and PSCL(4,4)-CRC(7,12,10,3), while adding the constraint in (22) results in PSCL(2,4)-CRC(20,12) and PSCL(4,4)-CRC(4,12,12,4). It is possible to see that the error-correction performance loss due to (22) is negligible for both $L = 2$ and $L = 4$. While these curves consider a code constructed for $E_b/N_0 = 2$ dB, similar results have been observed for any $E_b/N_0$ value.

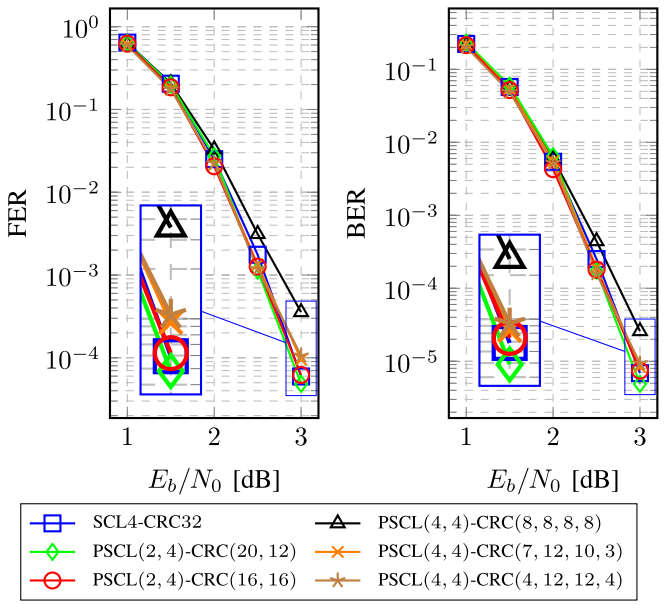


Fig. 10. FER and BER performance for PSCL decoding of $\mathcal{P}(1024, 512)$ when $L = 4$. The code is optimized for $E_b/N_0 = 2$ dB.

### *B. LLR-β Memory Sharing*

This Section presents a memory reduction technique that can be applied to decoders implementing an SC-based algorithm, like SC, Fast-SSC [11], and SCL (SCL-CRC). It does not imply any particular decoder hardware structure, since its basic idea is derived from the order with which calculations need to be performed according to SC. As described in Section II-A, the SC decoding process follows a specific operation schedule. This scheduling allows for substantial memory reduction for a SC decoder if the memory is shared between the LLR memory and the $\beta$ memory. Let us consider the example in Fig. 2. The vector of LLR values $\boldsymbol{\alpha}_2^{0\to3}$ is used to calculate the vectors $\boldsymbol{\alpha}_1^{0\to1}$ and $\boldsymbol{\alpha}_1^{2\to3}$. The vector of $\beta$ values $\boldsymbol{\beta}_2^{0\to3}$ is only created after both vectors $\boldsymbol{\alpha}_1^{0\to1}$

and $\boldsymbol{\alpha}_1^{2\to 3}$ are created. Since the vector $\boldsymbol{\alpha}_2^{0\to 3}$ is no longer needed, the vector $\boldsymbol{\beta}_2^{0\to 3}$ can use the same memory allocated for $\boldsymbol{\alpha}_2^{0\to 3}$. This will result in memory sharing between LLR and $\beta$ memories and subsequently results in memory saving. Since $\beta$ values are represented with one bit and LLR values with $Q_{\alpha_I}$ bits, we can store the $\beta$ values in the sign bit of the LLR values. The resulting memory requirement for the memory-reduced (MR) SC decoder is

$$M_{\mathrm{SC_{MR}}} = NQ_{\alpha_C} + (N-1)\,Q_{\alpha_I}, \tag{23}$$

which has $N-1$ fewer memory bits than (8).

SCL decoders follow the same schedule as SC decoders and thus the LLR and $\beta$ memory sharing can be applied to them. Following the same reasoning for SC decoders, the memory requirement of the MR SCL decoder can be calculated as

$$M_{\mathrm{SCL_{MR}}} = NQ_{\alpha_C} + L\,(N-1)\,Q_{\alpha_I} + LQ_{\mathrm{PM}} + LN. \tag{24}$$

It can be seen that the $LN$ memory bits required to store the final candidate codewords are present in the MR SCL decoder and MR SCL requires $(N-1)L$ fewer memory bits than a conventional SCL decoder.

The PSCL decoder uses the SCL-CRC decoder to decode the partitions and uses SC decoding rules to pass the candidate codewords from one partition to another. Therefore, the LLR-$\beta$ memory sharing can be applied as well, to the SCL-CRC and the SC decoders both. The resulting memory requirement for the MR PSCL decoder is

$$M_{\mathrm{PSCL_{MR}}} = NQ_{\alpha_C} + \left(\sum_{k=1}^{\log_2 P}\frac{N}{2^k} + L\left(\frac{N}{P}-1\right)\right)Q_{\alpha_I} + LQ_{\mathrm{PM}} + L\frac{N}{P}. \tag{25}$$

The proposed technique can be applied to all SC-based decoding algorithms. No approximation is used, and it incurs no error-correction performance degradation.

### C. Quantization Reduction for Channel LLRs

The channel LLR memory requires the storage of $N$ LLR values received from the channel. It was observed in [10] that quantizing channel LLR values with the same number of bits as the internal LLR values would not introduce significant error-correction performance loss in comparison with its floating point counterpart. For a code of length $N = 1024$, the channel LLR and the internal LLR values were quantized with 6 bits. The PM requires more bits than both channel and internal LLR values for minimal error-correction performance degradation due to quantization. Therefore, 8 bits were assigned to PM values. The same quantization scheme was adopted in the LLR-based SCL decoders of [12], [16]. In [13], [21], it was observed that the channel LLR values require fewer number of quantization bits and for a polar code of length $N = 1024$, the channel LLR values were quantized with 5 bits. As can be seen in Fig. 11 and Fig. 12, our simulations for $\mathcal{P}(1024, 512)$ show that setting $Q_{\alpha_C} = 4$ is adequate to keep the error-correction performance of SCL-CRC decoder close to the floating point version of channel LLR values for $L = 2$ and $L = 4$, for BER and FER values useful for wireless communications.

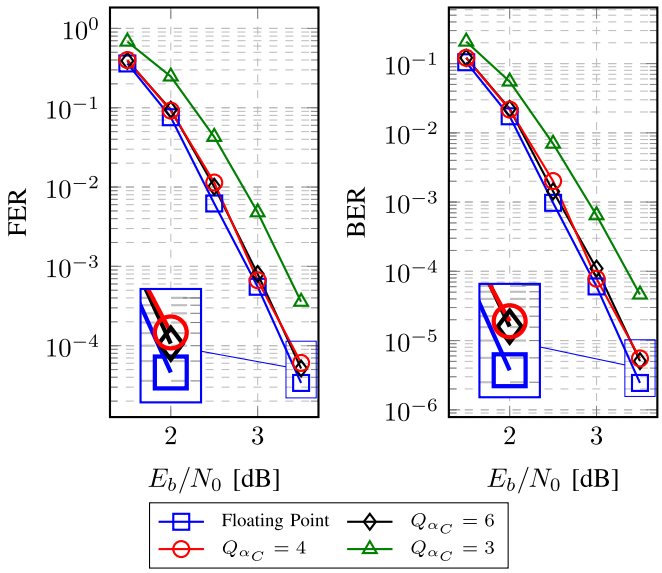


Fig. 11. Effect of $Q_{\alpha_C}$ on FER and BER performance for SCL-CRC decoding of $\mathcal{P}(1024, 512)$ when $L = 2$, $Q_{\alpha_I} = 6$, and $Q_{\mathrm{PM}} = 8$. The code is optimized for $E_b/N_0 = 2$ dB and the CRC length is 32.

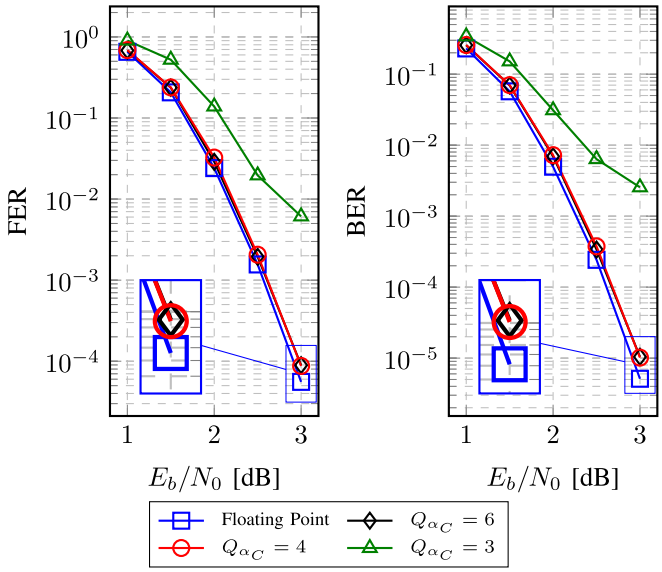


Fig. 12. Effect of $Q_{\alpha_C}$ on FER and BER performance for SCL-CRC decoding of $\mathcal{P}(1024, 512)$ when $L = 4$, $Q_{\alpha_I} = 6$, and $Q_{\mathrm{PM}} = 8$. The code is optimized for $E_b/N_0 = 2$ dB and the CRC length is 32.

### D. Low-Stage Memory Reduction for SRAM-Based SC Decoders

Most SC-based decoder architectures rely on a set of $P_e$ processing elements. Each of these modules performs (3) and (4) operations in parallel with the others: this helps to reduce decoding latency as long as the LLR vectors can be processed in few time steps. This concept has been applied to decoders implementing a variety of evolutions of the SC decoding algorithm [11], [13], [14], [21]–[26]. The decoder presented in [11] implements the Fast-SSC decoding algorithm, that reduces the latency of SC by pruning the tree. To limit the latency introduced by the computation of (3) and (4), the internal LLR and $\beta$ values for each stage of the SC decoding tree are stored concurrently. To accommodate

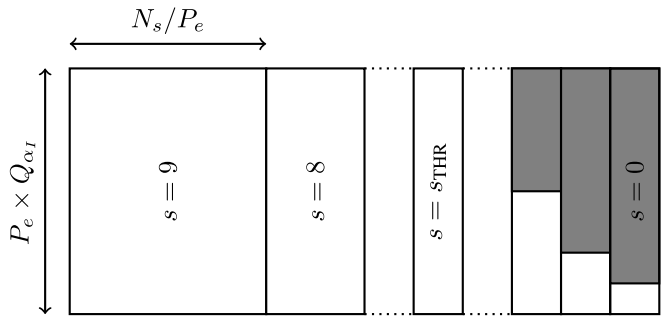


Fig. 13. SRAM-based memory architecture for $N = 1024$.

the parallelism degree of $P_e$ introduced by the processing elements, the LLR memory has a width of $P_e \times Q_{\alpha_I}$ bits, while the $\beta$ memory has a width of $P_e$ bits. Moreover, each memory is split into two single-port memory banks: words from two banks can be read and one memory bank can be written within the same clock cycle. To maintain this characteristic, a whole memory word is reserved for tree stages $s < s_{\text{THR}}$, where $s_{\text{THR}}$ is such that $2^{s_{\text{THR}}} = P_e$. Since for $s < s_{\text{THR}}$ the required memory is lower than $P_e$ for $\beta$ values and lower than $P_e \times Q_{\alpha_I}$ for LLR values, when memories are implemented as SRAM blocks there is an unused memory overhead. This is shown in Fig. 13, where the gray-colored memory blocks in stages $s < s_{\text{THR}}$ are unused. The LLR and $\beta$ memory requirements can be computed according to

$$M_{\text{SC}_{\text{SRAM}}} = P_e Q_{\alpha_I} \left( \sum_{k=\log_2 P_e}^{n-1} \frac{2^k}{P_e} + \log_2 P_e - 1 \right). \quad (26)$$

To avoid unnecessary memory instantiations without affecting decoder latency, it is useful to notice that the total amount of memory required by all the stages $s < s_{\text{THR}}$ is lower than that needed by the sole stage $s_{\text{THR}}$. This means that a single memory word of parallelism $P_e$ ($P_e \times Q_{\alpha_I}$) can be used to accommodate all the $\beta$ (LLR) values for stages $s < s_{\text{THR}}$. However, since the whole lower stage memory is contained within a single word of one of the memory banks, the single-clock-cycle read-during-write approach implemented by this type of decoder cannot be executed. The same decoding speed can nevertheless be maintained by duplicating the memory word allocated for $s < s_{\text{THR}}$. A memory controller selects which of the words to read from and write to at each clock cycle. The architecture is depicted in Fig. 14. This reduces the memory overhead to two bits. The new memory requirements are computed as

$$M_{\text{SC}_{\text{SRAM}}} = P_e Q_{\alpha_I} \left( \frac{N}{P_e} + 1 \right). \quad (27)$$

## IV. Hardware Implementation

### A. SC Decoder

The impact of the low-stage memory reduction technique described in Section III-D has been evaluated on an SC decoder architecture. We considered the Fast-SSC architecture from [11]. The original work is implemented for a polar code of length $N = 2^{15}$ and $P_e = 256$. The architecture can support any rate with a given instruction set, which is loaded to an instruction memory only once. For this work, we modified the architecture in [11] to derive a regular SC architecture with no tree pruning, and to accommodate a code length of $N = 1024$ and $P_e = 64$. Some small modifications in the memory routing logic and the datapath are required to implement the new memory structure. In particular, the memory address is fixed to the last word in the memory when a node belongs to stage $s < s_{\text{THR}}$. Simple swapping logic switches between the two copies of the last word.

### B. SCL-CRC Decoder

As a proof of concept, we considered as a starting point the SCL-CRC decoder architecture described in [10], sized for a polar code with $N = 1024$, and able to decode any code rate. We modified it to implement both the channel LLR quantization reduction and LLR-$\beta$ memory sharing.

Implementation of the LLR quantization reduction is straightforward. The channel LLR memory width is reduced to fit the new quantization: since LLR values are represented with sign and magnitude, the magnitude of values read from memory is zero-padded to fit the internal quantization $Q_{\alpha_I}$.

The LLR-$\beta$ memory sharing requires a few modifications to the hardware architecture in [10]: it does not change the scheduling of operations, that remains as shown in Fig. 2. The decoder relies on $P_e$ processing elements that can perform (3) and (4) operations in parallel. Each processing element receives as inputs two LLR and one $\beta$ values. Thus, the $\beta$ memory in standard SCL-CRC requires to be read with a parallelism of $P_e$ values, and the LLR memory with a parallelism of $2P_e$ values. If the $\beta$ memory is used to store LLR signs as well, then it must allow for additional $2P_e$ values to be read concurrently: since the $\beta$ memory in [10] is composed of registers only, due to its irregular update pattern, this modification can be achieved with dedicated multiplexers, as shown in Fig. 15. The update structure of the $\beta$ memory already allows for multiple irregular updates: in the standard SCL-CRC decoder structure, whenever $s = 0$ and a bit is estimated, all the $\beta$ values at all stages that are influenced by that bit are concurrently updated. The $P_e$ concurrent LLR signs can thus be easily stored in the $\beta$ memory by modifying the write enable generation logic and allowing for updates also when $s \neq 0$.

The additional memory addressing logic is instantiated in parallel to the standard one, and does not lie on the system critical path. Thus, the implementation of this technique does not decrease the achievable frequency, and since it does not add any step to the decoding process, the decoder latency and throughput remain unchanged. Moreover, control signals are already present in the standard SCL decoder architecture: thus, no additional logic is required to create them. The schedule described in Section III-B assures that newly computed $\beta$ values overwrite obsolete LLR signs, and updated LLR signs overwrite $\beta$ values that are no longer needed.

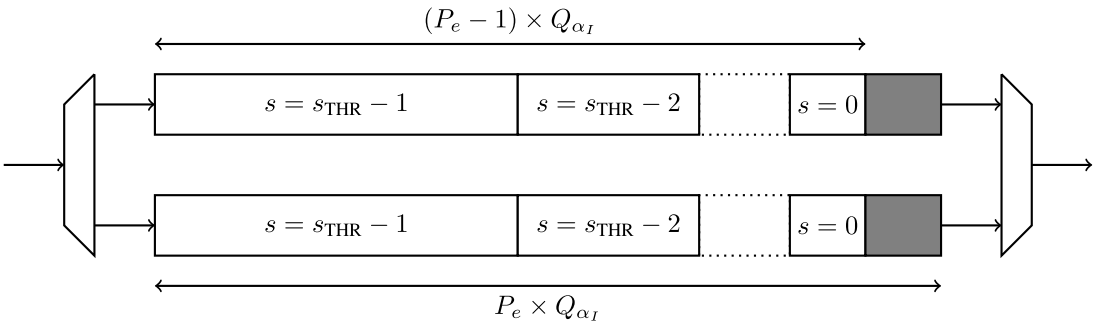


Fig. 14. Low-stage memory architecture.

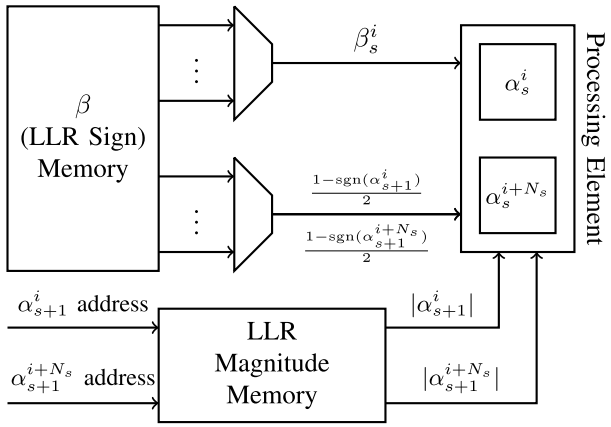


Fig. 15. LLR-$\beta$ memory sharing architecture.

### C. PSCL Decoder

To implement PSCL, the SCL-CRC decoder described in the previous Section have to undergo some architectural modification. Both LLR and $\beta$ memories have to be reduced to fit the size of the partition on which SCL-CRC decoding is performed. A secondary partial memory is instantiated to fit the part of the decoding tree that is handled by SC decoding, along with the related addressing and enabling logic.

On top of the PSCL decoder, we also applied the architectural modifications necessary to implement the LLR-$\beta$ memory sharing and channel LLR quantization reduction. These modifications are analogous to those described in Section IV-B for SCL-CRC. Applying the channel LLR quantization reduction remains straightforward, and the LLR sign and $\beta$ addressing logic is made simpler by the smaller LLR and $\beta$ memories.

## V. Results

The SC decoder described in Section IV-A has been described in VHDL and synthesized on an Altera Stratix IV EP4SGX530KH40C2 FPGA, with and without the SRAM reduction technique described in Section III-D. Table III reports the synthesis results in terms of look-up tables (LUTs), registers, RAM bits and achievable frequency. It can be seen that the proposed technique saves 11% of RAM bits, slightly reducing both sequential and combinational resource usage, with no degradation on the achievable frequency. Improvements of the same entity can be expected in case the method is applied to ASIC implementations.

TABLE III
FPGA Implementation Results for SRAM-Based SC Decoder for $\mathcal{P}(1024, 512)$ When $P_e = 64$

| | No Memory Reduction | SRAM Compression |
|---|---|---|
| LUTs | 4767 | 4733 |
| Registers | 1581 | 1532 |
| RAM [bits] | 35604 | 31764 |
| Frequency [MHz] | 158.83 | 160.23 |

The SCL-CRC and PSCL architectures described in Section IV-B and IV-C have been described in VHDL and synthesized in 65 nm TSMC CMOS technology. Table IV reports the synthesis results for architectures sized for $\mathcal{P}(1024, 512)$, targeting a frequency of 800 MHz, with all memory elements implemented with registers and with $P_e = 64$. Two SCL-CRC architectures are considered, for $L = 2$ and $L = 4$, both with a CRC of length 32 bits. PSCL architectures are considered for $L = 2$ and $L = 4$ as well, taking in account different numbers of partitions $P$ and different CRC lengths and distributions. All presented SCL-CRC and PSCL decoders rely on the same decoding flow and have been synthesized for the same target frequency of 800 MHz: they all yield a throughput of 301 Mb/s.

The first set of results (second and third column) details the total area occupation and the memory area occupation for the standard architectures, where no additional memory reduction technique is applied. Both channel and internal LLR values are quantized with 6 bits in all considered designs. We have designed and implemented our own decoders to be able to modify the architectures, implementing the memory reduction techniques. This allows us to correctly evaluate benefits and costs of the proposed methods. Nevertheless, the designed SCL-CRC decoders yield an area occupation very similar to that of state-of-the-art decoders when scaled to the same technology node. The work in [13] occupies 0.99 mm$^2$ for $L = 4$, while the area occupation of [21] is 1.03 mm$^2$ and 2.00 mm$^2$, respectively: our baseline SCL-CRC decoder has an area of 0.58 mm$^2$ for $L = 2$ and 0.99 mm$^2$ for $L = 4$. The SCL decoder in [22] occupies an area of 0.62 mm$^2$ for $L = 4$, and the SCL decoder described in [25] has an area of 0.73 mm$^2$ for $L = 4$. Most decoders in literature are single-code decoders built for high-throughput: on the other hand, our baseline SCL-CRC decoder is inspired to the high degree of

TABLE IV

SYNTHESIS AREA RESULTS WITH 65 NM TSMC CMOS TECHNOLOGY FOR SCL-CRC AND PSCL DECODING OF $\mathcal{P}(1024, 512)$. THE TARGET FREQUENCY IS 800 MHZ AND $P_e = 64$

| Algorithm | No Memory Reduction | | LLR-$\beta$ Sharing | | LLR-$\beta$ Sharing + Channel LLR Quantization | |
|---|---|---|---|---|---|---|
| | Total [mm$^2$] | Memory [mm$^2$] | Total [mm$^2$] | Memory [mm$^2$] | Total [mm$^2$] | Memory [mm$^2$] |
| SCL2-CRC32 | 0.5756 | 0.2774 | 0.4470 | 0.1934 | 0.4243 | 0.1868 |
| SCL4-CRC32 | 0.9888 | 0.4749 | 0.9002 | 0.3400 | 0.8768 | 0.3040 |
| PSCL(2,2)-CRC(16,16) | 0.4736 | 0.2097 | 0.3942 | 0.1844 | 0.3670 | 0.1785 |
| PSCL(4,2)-CRC(8,8,8,8) | 0.4348 | 0.1843 | 0.3695 | 0.1792 | 0.3400 | 0.1669 |
| PSCL(4,2)-CRC(4,12,12,4) | 0.4531 | 0.1866 | 0.3662 | 0.1793 | 0.3318 | 0.1569 |
| PSCL(2,4)-CRC(20,12) | 0.7548 | 0.3578 | 0.7411 | 0.3094 | 0.7273 | 0.2839 |
| PSCL(2,4)-CRC(16,16) | 0.7551 | 0.3524 | 0.7370 | 0.2999 | 0.7298 | 0.2842 |
| PSCL(4,4)-CRC(8,8,8,8) | 0.6998 | 0.2948 | 0.6786 | 0.2756 | 0.6503 | 0.2565 |
| PSCL(4,4)-CRC(4,12,12,4) | 0.7000 | 0.2947 | 0.6829 | 0.2741 | 0.6565 | 0.2581 |

flexibility of the decoder in [10], that can decode any code rate. The PSCL decoder implementations are direct modifications of the aforementioned SCL-CRC decoders. It is possible to see that PSCL decoders yield substantially lower total and memory area occupation with respect to the SCL-CRC decoders with the same list size $L$, with total area saving reaching 25% for $L = 2$ and 29% for $L = 4$. The memory reduction is more substantial as $P$ increases.

The effects on the area occupation brought by the LLR-$\beta$ sharing memory reduction technique detailed in Section III-B are reported in the fourth and fifth column of Table IV. Applied to the SCL-CRC decoder, it allows for 22% total area reduction when $L = 2$, and 9% when $L = 4$. The LLR-$\beta$ memory sharing technique is able to reduce the PSCL decoder memory by 12% in case of PSCL(2,2) and by 22% for PSCL(4,2), accounting for a combined total area saving of 32% and 36% reduction with respect to SCL-CRC. The LLR-$\beta$ area saving contribution for PSCL(2,4) and PSCL(4,4) accounts for around 3% of the total. Unlike PSCL, whose gain with respect to SCL-CRC increases with higher values of $L$, the impact of LLR-$\beta$ memory sharing decreases as $L$ increases. This is due to the total memory requirements of SCL-CRC decoders, that rise proportionally to the list size $L$.

The sixth and seventh columns of Table IV report the area occupation for the different decoders, after the conjunct application of LLR-$\beta$ memory sharing and reduction of channel LLR quantization. Their implementation results in 26% and 11% total area reduction for SCL-CRC with $L = 2$ and $L = 4$ respectively. The area reduction brought by these two techniques can reach 24% and 27% in case of PSCL(2,2) and PSCL(4,2), while it settles around 3% and 7% for PSCL(2,4) and PSCL(4,4). The combined contribution of PSCL, LLR-$\beta$ memory sharing and channel LLR quantization reduction leads to 42% and 34% total area saving with respect to SCL2-CRC32 and SCL4-CRC32, with memory area reduction peaking at 46% for PSCL(4,4)-CRC(4,12,12,4).

The total area reduction brought by each of the proposed memory reduction techniques with respect to the SCL-CRC benchmark decoders is summarized in Table V, for different values of $P$ and $L$. The results we provided target ASIC implementations, where memory plays a major role not only in terms of area occupation, but also in power consumption and energy efficiency. In case of FPGA implementation, the proposed memory reduction techniques will lead to gains in resource utilization proportional to those shown in Table V. However, both LUTs and registers contribute relatively less to the system power consumption than in the ASIC case. Thus, the proposed methods will lead to a lower energy efficiency improvement.

TABLE V

TOTAL AREA REDUCTION IN COMPARISON WITH SCL-CRC CONSIDERING DIFFERENT MEMORY REDUCTION TECHNIQUES

| Memory Reduction Technique | $P$ | $L$ | Total Area Reduction |
|---|---|---|---|
| PSCL | 2 | 2 | 18% |
| | 2 | 4 | 24% |
| | 4 | 2 | 21% |
| | 4 | 4 | 29% |
| PSCL + LLR-$\beta$ Memory Sharing | 2 | 2 | 32% |
| | 2 | 4 | 25% |
| | 4 | 2 | 36% |
| | 4 | 4 | 31% |
| PSCL + LLR-$\beta$ Memory Sharing + Channel LLR Quantization | 2 | 2 | 36% |
| | 2 | 4 | 26% |
| | 4 | 2 | 42% |
| | 4 | 4 | 34% |

## VI. CONCLUSION

In this paper, we proposed a set of memory-reduction techniques for SC and SCL decoders. We have shown that these techniques are independent from the decoder architecture. We have verified the effectiveness of our memory-reduction approaches by implementing the decoders on hardware and the synthesis results show up to 46% memory reduction in comparison with benchmark SC-based decoders. Since memory is a dominant factor in the total area occupation of SC-based decoders, the memory-reduction techniques proposed in this paper reduce the total area occupation of SC-based decoders of up to 42%.

## ACKNOWLEDGMENT

The authors would like to thank Dr. Alexios Balatsoukas-Stimming for the original HDL code for SCL decoder with $N = 2048$ and $L = 4$ used in [16], from which the baseline benchmark decoders in this work are derived.

## REFERENCES


[1] E. Arıkan, "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. 2009.

[2] Reno, NV, USA. 3GPP. (Nov. 2016). *Final Report of 3GPP TSG RAN WG1 #87 v1.0.0*. http://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_87/Report/Final_Minutes_report_RAN1%2387_v100.zip

[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] G. Sarkis, "Efficient encoders and decoders for polar codes: Algorithms and implementations," Ph.D. dissertation, Dept. Elect. Comput. Eng., McGill Univ., Montreal, QC, Canada, Apr. 2016.

[5] B. Yuan and K. K. Parhi, "Belief propagation decoding of polar codes using stochastic computing," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2016, pp. 157–160.

[6] J. Sha, J. Lin, and Z. Wang, "Stage-combined belief propagation decoding of polar codes," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2016, pp. 421–424.

[7] Q. Zhang, A. Liu, X. Pan, and Y. Zhang, "Symbol-based belief propagation decoder for multilevel polar coded modulation," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 24–27, Jan. 2017.

[8] S. M. Abbas, Y. Fan, J. Chen, and C.-Y. Tsui, "High-throughput and energy-efficient belief propagation polar code decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 3, pp. 1098–1111, Mar. 2017.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] C. Xiong, J. Lin, and Z. Yan, "A multimode area-efficient SCL polar decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 12, pp. 3499–3512, Dec. 2016.

[14] J. Lin and Z. Yan, "An efficient list decoder architecture for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2508–2518, Nov. 2015.

[15] S. A. Hashemi, C. Condo, and W. J. Gross, "List sphere decoding of polar codes," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 1346–1350.

[16] S. A. Hashemi, A. Balatsoukas-Stimming, P. Giard, C. Thibeault, and W. J. Gross, "Partitioned successive-cancellation list decoding of polar codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2016, pp. 957–960.

[17] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.

[18] S. A. Hashemi, M. Mondelli, S. H. Hassani, R. Urbanke, and W. J. Gross, "Partitioned list decoding of polar codes: Analysis and improvement of finite length performance," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017. [Online]. Available: http://arxiv.org/abs/1702.06901

[19] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.

[20] *Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding*, document 3GPP TS 36.212 RAN #76 v14.3.0, 3GPP, West Palm Beach, FL, USA, Jun. 2017. [Online]. Available: http://www.3gpp.org/ftp//Specs/archive/36_series/36.212/36212-e30.zip

[21] J. Lin, C. Xiong, and Z. Yan, "A high throughput list decoder architecture for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2378–2391, Jun. 2016.

[22] B. Yuan and K. K. Parhi, "LLR-based successive-cancellation list decoder for polar codes with multibit decision," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 1, pp. 21–25, Jan. 2017.

[23] 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.

[24] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 318–328, Feb. 2016.

[25] C. Xiong, J. Lin, and Z. Yan, "Symbol-decision successive cancellation list decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 64, no. 3, pp. 675–687, Feb. 2016.

[26] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2268–2280, Oct. 2015.


**Seyyed Ali Hashemi** (S'16) was born in Qaemshahr, Iran. He received the B.Sc. degree in electrical engineering from Sharif University of Technology of Technology, Tehran, Iran, in 2009 and the M.Sc. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 2011. He is currently pursuing the Ph.D. degree in electrical and computer engineering with McGill University, Montréal, QC, Canada. He was the recipient of a Best Student Paper Award at the 2016 IEEE International Symposium on Circuits and Systems. His research interests include error-correcting codes, hardware architecture optimization, and VLSI implementation of digital signal processing systems.

**Carlo Condo** (M'15) received the M.Sc. degree in electrical and computer engineering from Politecnico di Torino and University of Illinois at Chicago in 2010. He received his Ph.D. degree in electronics and telecommunications engineering from Politecnico di Torino and Telecom Bretagne in 2014. Since 2015, he has been a Post-Doctoral Fellow with the ISIP Laboratory, McGill University, and since 2017, he has been the McGill University Delegate with the 3GPP meetings for the 5th generation wireless systems standard. His Ph.D. thesis was awarded a mention of merit as one of the five best of 2013/2014 by the GE association, and he has been the recipient of two conference best paper awards, at SPACOMM 2013 and ISCAS 2016. His research is focused on channel coding, design and implementation of encoder and decoder architectures, digital signal processing, and machine learning.

**Furkan Ercan** (S'11) received the B.Sc. degree in electrical and electronics engineering and the M.Sc. degree in sustainable environment and energy systems from the Middle East Technical University, Northern Cyprus Campus, Ankara, Turkey, in 2011 and 2015, respectively. From 2011 to 2012, he was a Full Time Research and Development Intern with Intel Corporation, Hillsboro, OR, USA, focusing on system level energy efficiency on enterprise platforms. He is currently pursuing a Ph.D. degree with McGill University, Montréal, QC, Canada. His research interests are algorithm, design and implementation of signal processing systems with a focus on polar codes, and energy aware hardware architectures. He received a Best Student Paper Award in 2015 IEEE International Conference in Energy Aware Computing, Cairo, Egypt.

**Warren J. Gross** (S'92–M'04–SM'10) received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1999 and 2003, respectively.

He is currently Professor and Associate Chair (Academic Affairs) with the Department of Electrical and Computer Engineering, McGill University, Montréal, QC, Canada. His research interests include the design and implementation of signal processing systems and custom computer architectures. He served as Chair of IEEE Signal Processing Society Technical Committee on Design and Implementation of Signal Processing Systems, General Co-Chair of IEEE GlobalSIP 2017, and IEEE SiPS 2017, as Technical Program Co-Chair of SiPS 2012, an Organizer for the Workshop on Polar Coding in Wireless Communications at WCNC 2017, the Symposium on Data Flow Algorithms and Architecture for Signal Processing Systems (GlobalSIP 2014), IEEE ICC 2012 Workshop on Emerging Data Storage Technologies, Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and is currently Senior Area Editor. He is a Licensed Professional Engineer in the Province of Ontario.