PARTITIONED SUCCESSIVE-CANCELLATION LIST DECODING OF POLAR CODES

Seyyed Ali Hashemi[⋆]*, Alexios Balatsoukas-Stimming*[†]*, Pascal Giard*[⋆]*,*
Claude Thibeault[◇] *and Warren J. Gross*[⋆]

[⋆]McGill University, Montréal, Québec, Canada
[†]École polytechnique fédérale de Lausanne, Lausanne, Switzerland
[◇]École de technologie supérieure, Montréal, Québec, Canada

ABSTRACT

Successive-cancellation list (SCL) decoding is an algorithm that provides very good error-correction performance for polar codes. However, its hardware implementation requires a large amount of memory, mainly to store intermediate results. In this paper, a partitioned SCL algorithm is proposed to reduce the large memory requirements of the conventional SCL algorithm. The decoder tree is broken into partitions that are decoded separately. We show that with careful selection of list sizes and number of partitions, the proposed algorithm can outperform conventional SCL while requiring less memory.

***Index Terms*—** Partitioned List Decoder, Successive-Cancellation List Decoder, Polar Codes, Hardware Implementation.

1. INTRODUCTION

Polar codes provably achieve the symmetric capacity of memoryless channels and therefore have gained a lot of attention as promising error-correcting codes [1]. Successive-cancellation (SC) decoding was first proposed as a low-complexity decoding algorithm for polar codes. It was shown that the error probability of polar codes under SC decoding goes to zero as the blocklength goes to infinity, provided that the rate of the polar code is less than the capacity of the channel. From a hardware implementation point of view, SC decoding can be represented as a decoder tree having a fixed time and space complexity and is thus very attractive [2]. However, the algorithm is sub-optimal, especially for decoding moderate-length polar codes.

To improve the error-correction performance of SC decoding, the SC list (SCL) decoding algorithm was proposed in [3]. Unlike SC decoding, which estimates each bit based on the estimation of previous bits, SCL keeps a constrained list of the most likely candidates at each decoding step using the log-likelihood (LL) of each candidate. SCL reduces the gap between SC and maximum likelihood (ML) decoding at the cost of increased complexity. Furthermore, it was shown that concatenating polar codes with a cyclic redundancy check (CRC) as an outer code improves the performance of SCL to the extent where polar codes decoded with CRC-aided SCL are able to outperform low-density parity-check (LDPC) codes of the same length and rate [3]. To reduce the hardware complexity associated with LL-based SCL decoding, log-likelihood ratio (LLR) values were used and the path metric calculations adapted accordingly in [4]. Unfortunately, similarly to its LL-based counterpart, LLR-based SCL decoding requires a large memory to store the intermediate values, i.e. the total core area is often largely dominated by memory [4].

In this paper, a *partitioned* SCL (PSCL) decoding algorithm is proposed in order to reduce the memory requirements associated with SCL decoding. More specifically, PSCL decoding performs SCL decoding on partitions of the decoder tree and only one path candidate is transferred from one partition to the next. As a result, memory can be shared between the different partitions of the code, therefore, significantly reducing the overall memory requirements. Without loss of generality, we propose a CRC-aided scheme.

The paper is organized as follows: Section 2 offers a brief overview on polar encoding and decoding. Section 3 describes the proposed PSCL algorithm and compares its error-correction performance with that of conventional SCL decoding. In Section 4 hardware implementation results are presented showing memory and total area savings of up to 41% and 42%, respectively, at similar error-correction performance. Finally, conclusions are drawn in Section 5.

2. POLAR CODES

A polar code of length $N = 2^n$ which carries K information bits, denoted by $\mathcal{P}(N, K)$, has rate $R \triangleq \frac{K}{N}$ and is constructed by concatenating two polar codes of length $\frac{N}{2}$. Let us consider an input set $\mathbf{u}_0^{N-1} = \{u_0, u_1, \ldots, u_{N-1}\}$ and a coded set $\mathbf{x}_0^{N-1} = \{x_0, x_1, \ldots, x_{N-1}\}$. The recursive concatenation process can be expressed as a modulo-2 matrix multiplication as in

$$\mathbf{x}_0^{N-1} = \mathbf{u}_0^{N-1}\mathbf{G}^{\otimes n}, \tag{1}$$

where $\mathbf{G}^{\otimes n}$ is the n-th Kronecker product of the polarizing matrix $\mathbf{G} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$.

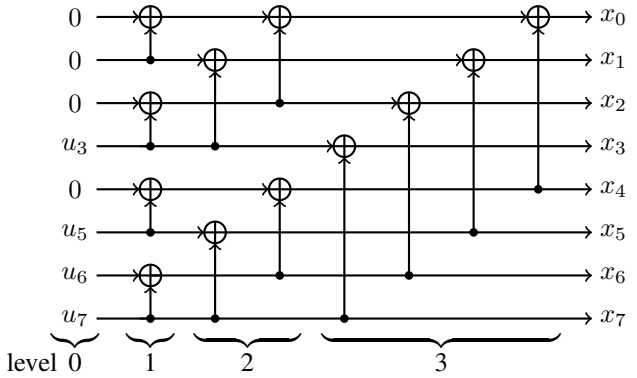

Fig. 1: Polar encoding for $\mathcal{P}(8,4)$.

Polar encoding consists of finding the K most reliable bit-channels and transmitting the information bits through them. The $N-K$ least reliable bits are set to a predefined value (usually 0) which is known by the decoder, and thus are called frozen bits. An example of polar encoding for $\mathcal{P}(8,4)$ is illustrated in Fig. 1, where $\mathbf{x}_0^{N-1}$ is generated by the encoder before being modulated and sent through the channel. The noisy channel output $\mathbf{y}_0^{N-1}$ is input to the polar decoder.

SC decoding provides each bit estimate $\hat{u}_i$ based on $\mathbf{y}_0^{N-1}$, the previously estimated bits $\hat{\mathbf{u}}_0^{i-1}$, and the location of frozen bits $\mathcal{F}$. The LLR-based formulation is

$$\hat{u}_i = \begin{cases} 0, & \text{if } i \in \mathcal{F} \text{ or } \log \frac{\mathrm{P}(\mathbf{y}_0^{N-1}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 0)}{\mathrm{P}(\mathbf{y}_0^{N-1}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 1)} \geq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

SC works on a decoder tree such as the one illustrated in Fig. 2. There are two types of messages passed through the different levels in a decoder tree: 1) the soft messages which contain the LLR values α; 2) the hard bit estimates β. Each node at level s of the decoder tree contains 2^s bits and the messages in Fig. 2 are calculated as

$$\begin{aligned} \alpha_l[i] =& \mathrm{sgn}(\alpha[i])\mathrm{sgn}(\alpha[i+2^{s-1}])\min(|\alpha[i]|, |\alpha[i+2^{s-1}]|), \\ \alpha_r[i] =& \alpha[i+2^{s-1}] + (1-2\beta_l[i])\alpha[i], \end{aligned} \tag{3}$$

and

$$\beta[i] = \begin{cases} \beta_l[i] \oplus \beta_r[i], & \text{if} \quad i < 2^{s-1} \\ \beta_r[i+2^{s-1}], & \text{otherwise,} \end{cases} \tag{4}$$

where $\oplus$ denotes the bitwise XOR operation [2] and $\beta[i]$ are called *partial sums*.

To improve the error-correcting performance of SC decoding, for each non-frozen bit, SCL decoding creates two tentative paths on the decoding tree corresponding to $\hat{u}_i = 0$ and $\hat{u}_i = 1$. In order to avoid an exponential growth in the number of tentative paths, only the L best (i.e., most likely) paths are kept. Specifically, in LLR-based SCL decoding [4], the L best paths are determined by the following path metric

$$\mathrm{PM}_i^l = \begin{cases} \mathrm{PM}_{i-1}^l, & \text{if } \hat{u}_i^l = \frac{1}{2}\left(1 - \mathrm{sgn}\left(\alpha_i^l\right)\right), \\ \mathrm{PM}_{i-1}^l + |\alpha_i^l|, & \text{otherwise,} \end{cases} \tag{5}$$

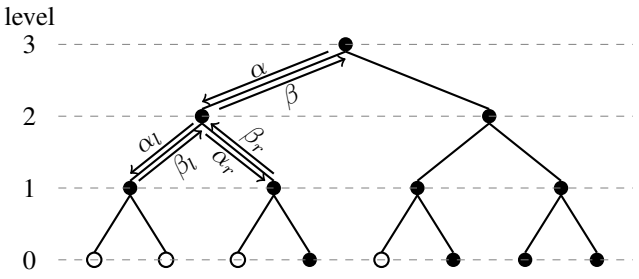

Fig. 2: SC decoder tree for $\mathcal{P}(8,4)$.

where l is the path index and α_i^l is the LLR value associated with the i-th bit at path l. A smaller path metric indicates a more reliable path.

Unfortunately, SCL decoding requires a large amount of memory to store the intermediate values. Let us assume that the LLR and path metric values are quantized with Q_α and Q_{PM} bits, respectively. The total memory requirements for the storage of the LLR values α, the path metrics PM, and the partial sum values β in the SC and SCL algorithms are [2]

$$M_{\mathrm{SC}} = \underbrace{(2N-1)\,Q_\alpha}_{\alpha \text{ (LLR values)}} + \underbrace{N-1}_{\beta \text{ (partial sums)}}, \tag{6}$$

and [4]

$$M_{\mathrm{SCL}} = \underbrace{(N+(N-1)\,L)\,Q_\alpha}_{\alpha \text{ (LLR values)}} + \underbrace{LQ_{\mathrm{PM}}}_{\text{path metrics}} + \underbrace{(2N-1)\,L}_{\beta \text{ (partial sums)}}, \tag{7}$$

respectively. We note that Q_{PM} grows at most as $\log N$ [4], so the term LQ_{PM} is negligible in Eq. (7).

3. PARTITIONED SCL DECODING OF POLAR CODES

The large memory requirements of the SCL algorithm translate into a large area occupation in the actual hardware decoder implementation. In fact, the total area is often largely dominated by memory, e.g. the memory area accounts for 45% of the total area in [4]. In order to reduce the required memory and, therefore, the area of the decoder, we propose a *partitioned* SCL (PSCL) decoding technique.

3.1. Proposed PSCL Decoding Algorithm

The conventional CRC-aided SCL decoding algorithm first performs SCL decoding to obtain the L most likely codeword candidates and, in the end, selects the (hopefully) correct estimate by choosing the candidate that matches the expected CRC. If no codeword verifies the CRC, the candidate with the best path metric is selected.

In PSCL decoding, on the other hand, the decoder tree is broken into partitions (i.e., subtrees) and SCL decoding is performed only on the partitions, while the standard SC rules

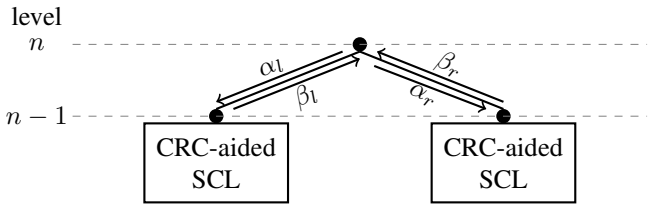

Fig. 3: PSCL with two partitions.

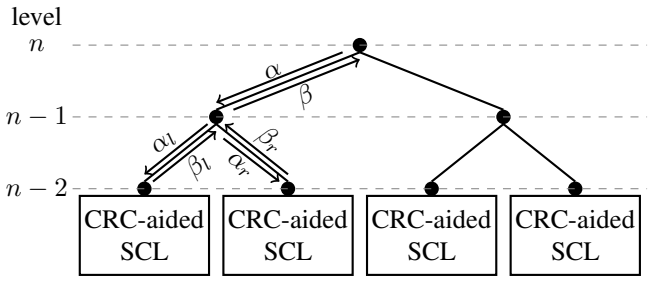

Fig. 4: PSCL with four partitions.

are applied to the remainder of the decoding tree. Each partition outputs a single candidate codeword which is selected with the help of a CRC and then sent to the next partition for further decoding. The decoding process starts with the standard SC update rules given by (3) and (4). Therefore, the decoder does not require memory to store L entire trees of internal LLRs, but only L copies of the partitions on which SCL decoding is performed.

Fig. 3 and Fig. 4 show the PSCL process when a code is broken into two and four partitions, respectively. The memory used in each CRC-aided SCL decoding block can be shared with the next decoding block since only one candidate survives after decoding each partition. The total memory usage in PSCL with P partitions and list size L can be calculated as

$$M_{\text{PSCL}} = \underbrace{\left(\sum_{k=0}^{P-1}\frac{N}{2^k} + \left(\frac{N}{2^{P-1}}-1\right)L\right)Q_\alpha}_{\alpha\text{ (LLR values)}} + \underbrace{LQ_{\text{PM}}}_{\text{path metrics}} + \underbrace{\sum_{k=1}^{P-2}\frac{N}{2^k} + \left(\frac{N}{2^{P-2}}-1\right)L}_{\beta\text{ (partial sums)}}, \quad (8)$$

where $P \geq 2$ and $P = 1$ makes PSCL decoding equivalent to conventional SCL decoding. Also note that when $P = 2$, $\sum_{k=1}^{P-2}\frac{N}{2^k} = 0$.

It should be noted that the lower bound on the memory usage for PSCL is the memory requirement of the SC algorithm and the upper bound is the memory required by SCL with list size L. Fig. 5 illustrates the PSCL memory usage with different numbers of partitions and list sizes for a polar code with $N = 2048$, $Q_\alpha = 6$ bits, and $Q_{\text{PM}} = 8$ bits.

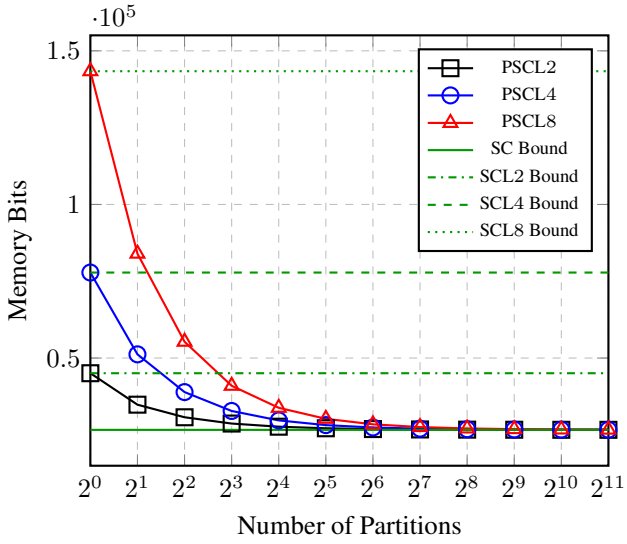

Fig. 5: Memory requirements for polar codes of length $N = 2048$. PSCLL (SCLL) denotes the PSCL (SCL) decoding algorithm with list size L.

As it can be seen in the figure, the amount of memory decays exponentially towards the SC bound as the number of partitions increases. In other words, a small increase in the number of partitions results in significant savings, e.g. using four partitions in PSCL4 is expected to require less memory than SCL2.

3.2. Error-Correction Performance

Fig. 6 shows the frame error rate (FER) and bit error rate (BER) performance of SCL and PSCL. The error-correction performance of the plain SC algorithm is also included as a reference. SCLL-CRCx denotes the SCL algorithm with list size L and CRC length x and PSCL(P, L)-CRCx represents the PSCL algorithm with P partitions, list size L, and a CRC of length x.

The performance results are provided for a polar code of length $N = 2048$ and rate $R = \frac{1}{2}$, while a CRC of length 32 is used for the conventional SCL decoding algorithm. To keep the code rate unchanged and to have a fair comparison, PSCL$(2, L)$ uses a CRC of length 16, i.e. each of its two partitions uses a CRC16. Similarly, for PSCL$(4, L)$ each of the four partitions uses a CRC8. The CRC polynomials were taken from [5, 6].

Fig. 6 shows that PSCL$(2, 2)$-CRC16 has identical FER and BER performance compared to SCL2-CRC32 and there is only a slight deterioration in performance when the code is broken into four partitions, as shown by the PSCL$(4, 2)$-CRC8 curve. However, in Fig. 6, it can also be seen that PSCL$(4, 4)$-CRC8 has superior error-correction performance compared to that of SCL2-CRC32. Furthermore, PSCL$(4, 4)$ actually requires slightly less memory than SCL2, as shown in Fig. 5. Thus, PSCL achieves better performance *and* reduces memory usage at the same time.

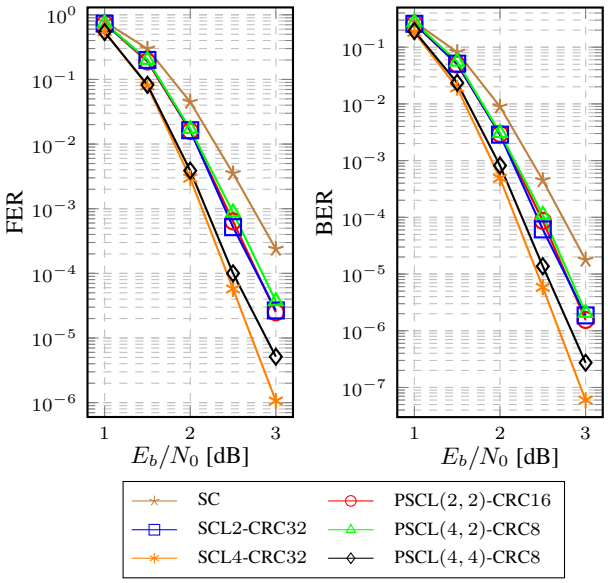

Fig. 6: Frame error rate (FER) and bit error rate (BER) performance comparison between CRC-aided SCL and PSCL decoding of $\mathcal{P}(2048, 1024)$. The code is optimized for $E_b/N_0 = 2$ dB.

4. HARDWARE IMPLEMENTATION RESULTS

Table 1 presents indicative synthesis results to compare SC, the conventional CRC-aided SCL algorithm with $L \in \{2, 4\}$, and the proposed PSCL algorithm for $L \in \{2, 4\}$ and $P \in \{2, 4\}$, for a polar code of blocklength $N = 2048$. For the CRC-aided SCL algorithm, the hardware architecture of [4] is used while an appropriately modified version of [4] was used for the PSCL algorithm. All synthesis results are for a TSMC 90 nm CMOS library (1 V, 25°C) with a target frequency of 500 MHz. All decoders have an equal latency of 5248 clock cycles (10.5 μs) and throughput of 164 Mbps.

From Table 1, we observe that the PSCL$(2, 2)$ and PSCL$(4, 2)$ decoders require 23% and 41% less memory area than the SCL2 decoder, respectively. The PSCL$(4, 4)$ decoder implementation is shown to require 23% less memory area than the SCL2 decoder while offering a better coding gain by approximately 0.25 dB at a BER of 10^{-5}. Regardless of the implementation, the memory area of the list decoders amounts to 40%–45% of the total area. The memory savings observed for the PSCL implementations thus translate into very significant reductions in the total area, making them very attractive compared to the conventional SCL decoders.

5. CONCLUSION

In this paper, we have proposed a novel partitioned list decoding algorithm for polar codes. In this algorithm, the code is broken into partitions and each partition is decoded with a CRC-aided successive-cancellation list decoder. Since the memory is shared between different partitions in the code, the memory requirements of a hardware implementation of partitioned list decoder is significantly smaller than that of a conventional list decoder without any error-correction performance degradation. Implementation results show that at equivalent error-correction performance, the proposed algorithm leads to memory and total area savings of 41% and 42%, respectively, when compared to a similar list decoder implementation. Moreover, the proposed algorithm enables a coding gain of approximately 0.25 dB at a bit error rate of 10^{-5} while occupying 13% less total area than the conventional CRC-aided successive-cancellation list decoder.

Table 1: Synthesis area results for the SC, CRC-aided SCL, and PSCL decoding algorithms.

Algorithm	Total (mm^2)	Memory (mm^2)
SC	0.723	0.413
SCL2-CRC32	1.563	0.702
SCL4-CRC32	3.075	1.214
PSCL$(2, 2)$-CRC16	1.189	0.540
PSCL$(4, 2)$-CRC8	0.909	0.415
PSCL$(4, 4)$-CRC8	1.356	0.543

ACKNOWLEDGEMENT

The authors would like to thank Gabi Sarkis and Carlo Condo of McGill University for helpful discussions.

6. 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, July 2009.

[2] 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.

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

[4] A. Balatsoukas-Stimming, M. Bastani 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.

[5] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *IEEE Int. Conf. on Dependable Syst. and Netw. (DSN)*, 2004, pp. 145–154.

[6] P. Koopman, "32-bit cyclic redundancy codes for internet applications," in *IEEE Int. Conf. on Dependable Syst. and Netw. (DSN)*, 2002, pp. 459–468.