doi: http://dx.doi.org/10.1109/ICCNC.2013.6504198

Polar Codes for Data Storage Applications

Gabi Sarkis and Warren J. Gross

Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada
Email: gabi.sarkis@mail.mcgill.ca and warren.gross@mcgill.ca

***Abstract*—Polar codes are a new class of error-correcting codes that provably achieve the capacity of memoryless channels with low complexity encoding and decoding algorithms. In this work, we survey the literature and investigate the suitability of polar codes to data storage applications focusing on error-correction performance and throughput. We show that polar codes meet the criteria for such applications and highlight the work required before practical data storage systems can utilize these codes.**

I. Introduction

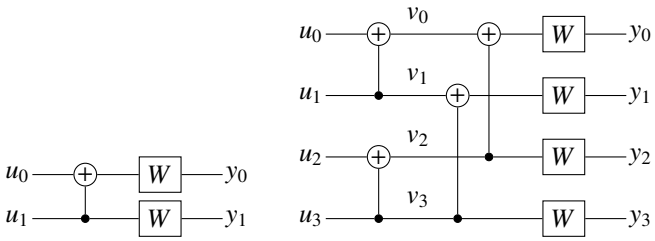
Error correction schemes for storage systems must satisfy three requirements: they must have good waterfall region performance, very low error-floors, and moderate throughput in the range of multiple gigabits per second (Gbit/s).

Polar codes, introduced in [1], are the first codes to provably achieve the capacity of the underlying channel with an explicit construction and a tractable-complexity decoding algorithm: the successive cancellation (SC) decoding algorithm. For a code of length N, SC decoding has complexity $O(N \log N)$. This low complexity makes polar codes an attractive error correction scheme.

In this paper we survey the literature and study the suitability of polar codes for data storage applications. Since polar codes have very low error floors due to their large stopping distances [2], we focus on the waterfall region performance and the decoding throughput.

While polar codes achieve the channel capacity, they do so asymptotically and in practice, SC decoding of codes of short or moderate lengths have worse error-correction performance in the waterfall region than LDPC codes of comparable lengths. The immediate solution to improving the performance is to exploit the low complexity of the decoder and use longer codes: in [3], a code of length $N = 2^{17}$ was implemented on a field-programmable gate array (FPGA). However, this method is inefficient as it was shown in [4] that the error-correction performance scales slowly with code length. List decoding [5] improves the performance of polar codes without changing the code length. Additionally, it was found that concatenating a polar code with an error detection code, such as a cyclic redundancy check (CRC), improves the performance of both SC [6] and list [7] decoding. Finally, switching to non-binary polar codes improves the error-correction performance significantly [8].

The SC decoding algorithm estimates bits sequentially; therefore it has low throughput. In [9], it was shown that the information throughput of an SC decoder is approximately $0.5Rf$, where R is the code rate and f the decoder clock frequency. Therefore, an SC decoder for a polar code of length 2^{15} bits and rate 0.9 running at 500 MHz will have an information throughput of approximately 225 Mbit/s, which is insufficient for data storage. Among the many throughput improving methods proposed in literature, simplified successive cancellation (SSC) [10] and simplified successive cancellation with maximum likelihood nodes (ML-SSC) [11] offer the largest improvement. For the aforementioned example, SSC and ML-SSC can achieve information throughput of 2.8 Gbit/s and 4.5 Gbit/s, respectively, while retaining the same number of processing units.



(a) $N = 2$ (b) $N = 4$

Fig. 1: Construction of polar codes of lengths 2 and 4

We start this work by reviewing polar codes and the SC decoding algorithm in Section II. In Section III, we review the error-correction performance of polar codes and the different methods by which it can be improved. We study throughput improving methods in Section IV and present concluding remarks and avenues for future work in Section V.

II. Review of Polar Codes

A. Construction of Polar Codes

Polar codes utilize the channel polarization phenomenon to achieve the channel capacity as the code length increases. Channel polarization occurs when one constructs from an independent set of N identical channels a set of N channels whose probability of error-free transmission approaches either 1 or 0.5 as $N \to \infty$. The construction for $N = 2$ is illustrated in Fig. 1a, where the ability of correct estimation decreases for the bit u_0, but increases for u_1 compared to the case where the bits are transmitted directly over the channel W. The polarization increases as N increases and for $N > 2$, the channels can be combined recursively as shown in Fig. 1b for $N = 4$. As $N \to \infty$, each bit's probability of being successfully

estimated, approaches 1, perfectly reliable, or 0.5, completely unreliable, and the proportion of reliable bits approaches the capacity of W. Since the reliability of each bit is known a priori, the k most reliable bits are used to transmit information and the remaining bits, called the frozen bits, are set to a known value, usually 0, forming an (N, k) polar code.

Polar codes can be represented using generator matrices: for $N = 2$ the generator matrix is $F = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$. Generator matrices for longer codes are obtained using Kronecker powers, $^{\otimes}$, of F so that the generator matrix for a code of length N is $F^{\otimes \log_2 N}$. For example, for $N = 4$, the generator matrix is

$$F^{\otimes 2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

It should be noted that in this work, we use the non-bit-reversed generator matrices, unlike in [1] where the rows of the generator matrix are rearranged by bit-reversing their indices.

B. The Successive-Cancellation Decoding Algorithm

It was proved in [1] that polar codes achieve the channel capacity when decoded using the SC decoding algorithm, which sequentially provides estimates $\hat{u}_i$, $0 \leq i < N$, of the information and frozen bits: given the channel information vector $\mathbf{y}$ and the previously decoded bits $\hat{u}_0$ to $\hat{u}_{i-1}$, denoted $\hat{\mathbf{u}}_0^{i-1}$, the SC decoder estimates $\hat{u}_i$ according to:

$$\hat{u}_i = \begin{cases} 0, & \text{if } \frac{\Pr[\mathbf{y}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 0]}{\Pr[\mathbf{y}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 1]} > 1; \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

The probabilities $\Pr[\mathbf{y}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 0]$ and $\Pr[\mathbf{y}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 1]$ can be calculated recursively. Using the values $\mathbf{v}_0^{N-1}$ shown in Fig. 1b, the likelihood values for $\hat{u}_0$ can be calculated as follows:

$$\begin{aligned} \Pr[\mathbf{y}|\hat{u}_0 = 0] =& \Pr[\mathbf{y}|\hat{v}_0 = 0] * \Pr[\mathbf{y}|\hat{v}_1 = 0] + \\ & \Pr[\mathbf{y}|\hat{v}_0 = 1] * \Pr[\mathbf{y}|\hat{v}_1 = 1] \quad (2) \\ \Pr[\mathbf{y}|\hat{u}_0 = 1] =& \Pr[\mathbf{y}|\hat{v}_0 = 0] * \Pr[\mathbf{y}|\hat{v}_1 = 1] + \\ & \Pr[\mathbf{y}|\hat{v}_0 = 1] * \Pr[\mathbf{y}|\hat{v}_1 = 0]. \quad (3) \end{aligned}$$

The likelihoods of $\hat{u}_1$ depend on the value of $\hat{u}_0$ and are calculated as the following when $\hat{u}_0 = 0$:

$$\Pr[\mathbf{y}, \hat{u}_0 = 0|\hat{u}_1 = 0] = \Pr[\mathbf{y}|\hat{v}_0 = 0] * \Pr[\mathbf{y}|\hat{v}_1 = 0] \quad (4)$$

$$\Pr[\mathbf{y}, \hat{u}_0 = 0|\hat{u}_1 = 1] = \Pr[\mathbf{y}|\hat{v}_0 = 1] * \Pr[\mathbf{y}|\hat{v}_1 = 1]; \quad (5)$$

and

$$\Pr[\mathbf{y}, \hat{u}_0 = 1|\hat{u}_1 = 0] = \Pr[\mathbf{y}|\hat{v}_0 = 1] * \Pr[\mathbf{y}|\hat{v}_1 = 0] \quad (6)$$

$$\Pr[\mathbf{y}, \hat{u}_0 = 1|\hat{u}_1 = 1] = \Pr[\mathbf{y}|\hat{v}_0 = 0] * \Pr[\mathbf{y}|\hat{v}_1 = 1] \quad (7)$$

when $\hat{u}_0 = 1$.

These equations are applied recursively until the inputs used are the likelihood values calculated from the channel output.

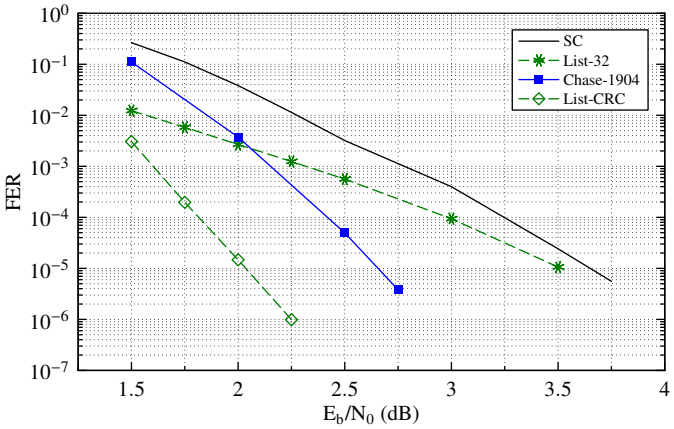


Fig. 2: FER of a (2048, 1024) polar coding system using different decoding algorithms

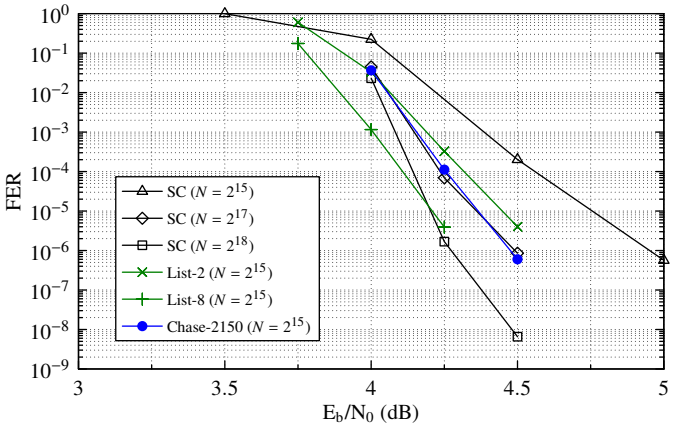


Fig. 3: FER of moderate length polar coding systems using different decoding algorithms

III. Error-Correction Performance

A. Successive-Cancellation Decoding

The frame error probability for polar codes and SC decoding decreases exponentially in the square root of the code length, i.e. it is $o(2^{-\sqrt{N}})$ [12]. This slow rate of decrease results in polar code having mediocre error-correction performance for codes of moderate length and limits the improvement gained by increasing the code length. Fig. 2 illustrates the performance of moderate length codes by showing the frame-error rate (FER) of SC decoding of a (2048, 1024) polar code. Fig. 3, which focuses on the issue of FER scaling with code length and demonstrates the performance of polar codes of rate 0.9, shows that increasing the code length from 2^{15} to 2^{17} improves the frame-error rate by 0.5 dB and further increasing the length to 2^{18} only yields a 0.25 dB improvement.

Despite its error-correction performance at moderate code lengths, SC decoding has an advantage in that its low-complexity enables the implementation of longer codes, e.g. in [3], a code of length 2^{17} was successfully implemented on an FPGA.

B. List Decoding

The first method in literature to have significantly improved the error-correction performance of polar codes without increasing their length was list decoding [13]. List decoding operates similarly to SC decoding; however, once an information bit is encountered, instead of making a decision as to what its value is, the decoder creates two paths: one in which the bit is assumed to be 0, and another in which it is 1. Left unconstrained, the decoder will generate 2^k paths; therefore, a maximum length, L, is imposed on the list. When the list contains L paths and a new information bit, $\hat{u}_i$, is encountered, the decoder will generate $2L$ paths that are then sorted according to their reliabilities—calculated from $\Pr[\mathbf{y}, \hat{\mathbf{u}}_0^{i-1} | \hat{u}_i = 0]$—and the L least reliable paths are discarded, maintaining the list size at L. At the end of the decoding process, the most reliable path is chosen and presented as the decoder output. When $L = k$, the decoder will be performing maximum-likelihood (ML) decoding.

The improvement due to list decoding can be observed in Fig. 3 for the (32768, 29491) code. When $L = 2$, the FER improves by 0.25 dB and approaches that for SC decoding and the 2^{17} code; and when $L = 8$, the improvement is 0.5 dB and the list decoder outperforms the SC decoder for the 2^{18} until FER = 10^{-6}.

For the (2048, 1024) code the FER for a list decoder with $L = 32$ is shown in Fig. 2, where it can be observed that it performs 0.75 dB better than the SC decoder when the FER = 10^{-2}; however, this advantage shrinks to 0.2 dB at FER = 10^{-5} due to the small minimum weight of polar codes [7]. This lead to the development of the List-CRC decoding algorithm.

C. List-CRC Decoding

One solution to the small minimum distance problem of polar codes is to concatenate them with another code. A CRC increases the minimum distance significantly and in [7] it was proposed that a CRC be used to improve the performance of list decoding.

In a list-CRC system, the information bits are first encoded using a CRC encoder then a polar encoder. The decoder is identical to a regular list decoder and only differs in the final decoder output selection: instead of immediately using the most reliable path, the decoder searches for a path that satisfies the CRC constraint; if one is found, it is presented as the decoder output; otherwise, the decoder outputs the most reliable path. While the absence of a path satisfying the CRC constraint indicates decoding failure; presenting the most reliable path improves the bit-error rate.

The FER improvement due to using list-CRC for the (2048, 1024) code is significant: it is 1 dB when FER = 10^{-3} and increases to 1.5 dB when FER = 10^{-5}. For that case, $L = 32$ and a 16-bit CRC was used.

D. Chase Decoding

The standard chase decoding algorithm consists of three parts: generating error patterns from different combinations of the least reliable bits, decoding the channel output while utilizing the error patterns, choosing the decoder output from the list of candidates. Chase decoding of polar codes [11] operates under the same principle. The least reliable bits in a polar code are known a priori and the error patterns can be generated offline. The information bits are encoded using a CRC first then a polar code, as in the list-CRC decoding algorithm, and the CRC is used as the criteria for the decoder output selection.

Decoding starts using a regular SC decoding assuming the all-zero error pattern, i.e. no errors occurred. If the CRC constraint is satisfied, decoding ends and the decoder outputs the current set of estimated bits $\hat{\mathbf{u}}_0^{N-1}$. However, if the CRC constraint is not satisfied, the most likely error pattern, as calculated offline, is selected and decoding restarts. If the currently decoded bit is in the error pattern, its value is flipped and decoding continues assuming the new value. If the CRC is still not satisfied, the next most likely pattern is selected and decoding restart. This process is repeated until the CRC constraint is satisfied or all the error-patterns have been tested, in which case the decoder outputs the estimated bits assuming the all-zero pattern.

Chase decoding of polar codes yields significant performance improvement for the (2048, 1024) code, where the gain relative to SC decoding was 0.75 dB when FER = 10^{-5} and the maximum number of attempts was 1904. The gain for the (32768, 29491) code was 0.5 dB when FER = 10^{-6} and the resulting performance was similar to that of the 2^{17} code. Like list-CRC decoding, chase decoding improves the slope of the error-rate curve and the performance gain relative to SC decoding increases as the SNR increases.

This method is effective because the likelihoods of the error-patterns decrease exponentially; therefore, the decoding process is only restarted a few times on average. For example, for the (32768, 29491) code shown in Fig. 3, the average number of decoding attempts was 1.03 when SNR = 4.5, indicating that the average throughput for the chase decoder was only 3% slower than that of the SC decoder.

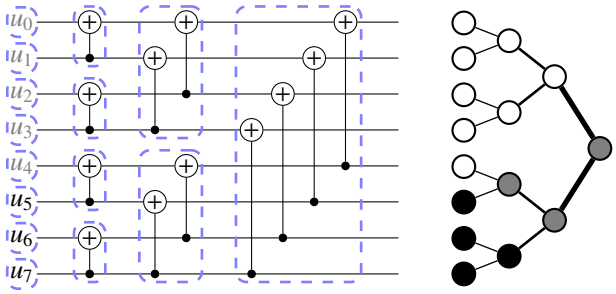
IV. Throughput

The sequential nature of SC decoding leads to low throughput decoders: the information throughput of an SC decoder is $k/(2N - 2)$ bit/s/Hz [9]. Once resource constraints are taken into account, the information throughput drops to

$$\frac{k}{2N + \frac{N}{P}\log_2(\frac{N}{4P})} \text{ bit/s/Hz},$$

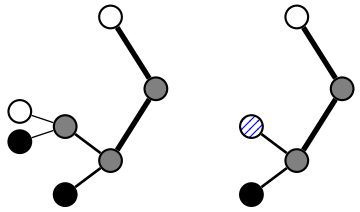
where P is the number of processing elements implemented in the semi-parallel (SP-SC) decoder [3]. It was shown in [3] that a small number (64) of processing elements was sufficient to achieve 90% of the throughput of unconstrained SC decoders for codes of length $< 2^{20}$. Therefore, increasing P does not improve the throughput significantly and one quickly approaches the limit of $k/(2N - 2)$.

While there have been multiple works in literature to improve throughput, SSC and ML-SSC are the only methods to improve it by an order of magnitude and approach the



(a) Standard (b) Tree

Fig. 4: Standard and tree representations of an SC polar decoder for an (8, 3) code.



(a) SSC (b) ML-SSC

Fig. 5: Decoder trees corresponding to the SC, SSC, and ML-SSC decoding algorithms

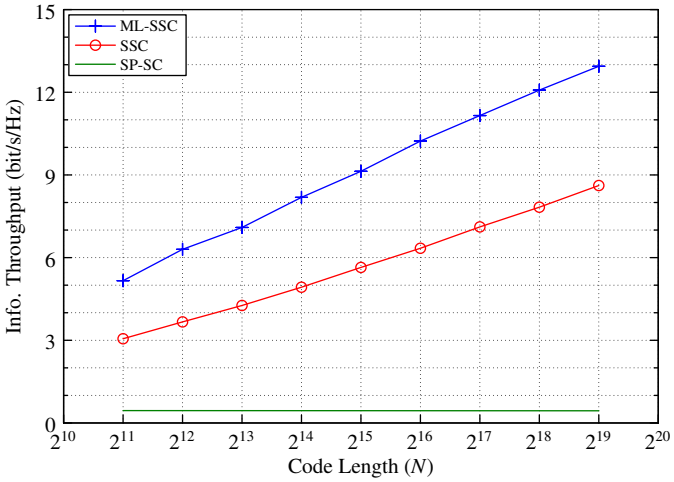


Fig. 6: Throughput of the ML-SSC, SSC, and SP-SC decoders for codes of different lengths and rate 0.9.

requirement for data storage. These two methods are reviewed in the following sections.

A. Simplified Successive-Cancellation Decoding

The SC decoder graph can be viewed as a tree, where each estimated bit $\hat{u}_i$ is a leaf node. There two types of leaf nodes: $\mathcal{N}^0$ nodes corresponding to frozen bits, and $\mathcal{N}^1$ nodes corresponding to information bits. The parent of two $\mathcal{N}^0$ nodes is also an $\mathcal{N}^0$ node and corresponds to a constituent polar code of rate zero. Combining two $\mathcal{N}^1$ nodes results in an $\mathcal{N}^1$ parent corresponding to a constituent code of rate one. Finally, the parent of two nodes of different types is an $\mathcal{N}^R$ node and corresponds to a constituent code of rate $0 < R < 1$. Fig. 4a and Fig. 4b demonstrate the transform for an (8, 3) polar code. The frozen bits have grey labels in Fig. 4a and $\mathcal{N}^0$, $\mathcal{N}^1$, and $\mathcal{N}^R$ nodes in Fig. 4b are in white, black, and grey, respectively.

Traversing the tree of Fig. 4b and calculating the output requires 14 clock cycles. However, the output of an $\mathcal{N}^0$ node is always the all-zero vector. $\mathcal{N}^1$ nodes correspond to rate one codes which do not improve the error rate according to the data-processing inequality; therefore, their output is simply the hard decision of the input soft-valued information. SSC [10] improves throughput by stopping the tree traversal and using the all-zero vector when an $\mathcal{N}^0$ node is encountered, and by stopping the tree traversal and using the hard-decision rule when an $\mathcal{N}^1$ node is encountered. This results in the pruned tree shown in Fig. 5a, which requires 9 instead of 14 clock cycles to decode.

The information throughput of SC and SSC decoders with the same number of processing elements, $P = 256$, for codes of rate $R = 0.9$ and lengths ranging from 2^{11} to 2^{19} is shown in Fig. 6, where it can be observed that SSC decoding is 6 to 18 times faster than SC decoding, with a throughput increasing from 3 to 8.5 bit/s/Hz as the code length increases.

B. Simplified Successive-Cancellation with ML Nodes

The main source of latency in SSC decoding are $\mathcal{N}^R$ nodes: they can require multiple clock cycles to calculate the data passed to their children, and they are the root of a tree which needs to be traversed. In [11], resource-constrained exhaustive-search ML decoding was used to decode the constituent codes corresponding to some $\mathcal{N}^R$ nodes, reducing latency and improving throughput. Exhaustive-search ML decoding was chosen because it eliminates the need for the sub-tree traversal and provides its output in one clock cycle. Therefore, an $\mathcal{N}^{\text{ML}}$ node replaced a regular $\mathcal{N}^R$ node when the following constraint is satisfied:

$$(2^{k_v} + 1)(n_v - 1) \leq P, \tag{8}$$

where k_v and n_v are the dimension and length of the constituent code, respectively, and P is the number of processing elements available. The ML-SSC decoder graph for the (8, 3) code is shown in Fig. 5b and requires 7 clock cycles to decoder a received channel vector.

Fig. 6 shows that the information throughput of ML-SSC decoding with $P = 256$ for codes of rate 0.9 varies from 5 to 13 bit/s/Hz—10 to 26 times the throughput of SC decoding, bringing the throughput of an ML-SSC decoder running at 500 MHz to 4.5 and 6 Gbit/s for the 2^{15} and the 2^{18} polar codes, respectively. The information throughput for these two codes can be increased to 5.3 and 7.6 Gbit/s by doubling the number of available processing elements to 512.

V. Conclusion

In this work we surveyed the literature investigating the suitability of polar codes for data storage applications. In terms

of error-correction performance, polar codes do not exhibit an error floor in the error-rate region of interest and their performance in the waterfall region can be improved using different methods, the two most effective being list-CRC and Chase decoding. Using SSC and ML-SSC, the throughput of polar decoder can be push into a suitable range. Future work will focus on combining error-correction performance and throughput improvement techniques. Finally, the memory effects of the channel need be studied and a solution proposed. Once these two remaining questions are resolved, we believe that polar codes will prove to be suitable for data storage systems.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] A. Eslami and H. Pishro-Nik, "On bit error rate performance of polar codes in finite regime," in *Proc. 48th Annual Allerton Conf. Communication, Control, and Computing (Allerton)*, 2010, pp. 188–194.

[3] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *submitted to IEEE Trans. Signal Processing*, 2012.

[4] S. H. Hassani and R. Urbanke, "On the scaling of polar codes: I. the behavior of polarized channels," in *Proc. (ISIT) Symp. IEEE Int Information Theory*, 2010, pp. 874–878.

[5] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. (ISIT) Symp. IEEE Int Information Theory*, 2011, pp. 1–5.

[6] G. Sarkis, C. Leroux, and W. J. Gross, "Chase decoding of polar codes," *submitted to IEEE Commun. Lett.*, 2012.

[7] I. Tal and A. Vardy, "List decoding of polar codes," *ArXiv e-prints*, 2012. [Online]. Available: http://arxiv.org/abs/1206.0050v1

[8] R. Mori and T. Tanaka, "Non-binary polar codes using reed-solomon codes and algebraic geometry codes," in *Proc. IEEE Information Theory Workshop (ITW)*, 2010, pp. 1–5.

[9] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in *Proc. IEEE Int Acoustics, Speech and Signal Processing (ICASSP) Conf*, 2011, pp. 1665–1668.

[10] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, 2011.

[11] G. Sarkis and W. J. Gross, "Increasing the throughput of polar decoders," *submitted to IEEE Commun. Lett.*, 2012.

[12] E. Arikan and E. Telatar, "On the rate of channel polarization," in *Proc. IEEE Int. Symp. Information Theory ISIT 2009*, 2009, pp. 1493–1495.

[13] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. (ISIT) Symp. IEEE Int Information Theory*, 2011.