Polar codes: A pipelined implementation

Erdal Arıkan
Bilkent University, Ankara, Turkey
arikan@ee.bilkent.edu.tr

***Abstract*—Polar codes are a class of codes that can achieve the capacity of binary-input memoryless channels with certain symmetries. These codes have a recursive structure that make it possible to encode and decode them within complexity $\mathcal{O}(N \log N)$ for a code of block length N. This paper presents pipelined architectures with identical modules that are useful for low-complexity implementation of polar codes both in hardware and software. The uniform structure of the modules in the design make it possible to trade complexity for time in hardware implementations.**

***Index Terms*—Polar codes, belief propagation decoding, error-correcting codes, Reed-Muller codes, iterative decoding.**

I. Introduction

POLAR coding is a code construction method that can achieve the capacity of symmetric binary-input discrete memoryless channels such as the binary symmetric channel (BSC) and binary erasure channel (BEC). This technique was introduced and theoretically analyzed in [1]. Some experimental results were presented in [2]. However, the details of polar code construction and efficient methods for encoder and decoder implementation have not been discussed in previous work. The aim of this paper is to address these issues.

Let $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $F^{\otimes n}$ denote the nth Kronecker power of F. For example,

$$F^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{1}$$

For any $N = 2^n$, $n \geq 1$, and $1 \leq K \leq N$, an (N, K) polar code is a block code whose generator matrix is a $K \times N$ submatrix of $F^{\otimes n}$ constructed in accordance with certain selection rules. Such rules were discussed in [2] and will be discussed further here.

To gain a wider perspective on polar codes, it will be beneficial to consider the class of all (N, K) codes with generator matrices that are arbitrary (K, N) submatrices of $F^{\otimes n}$. We will denote this class by $\mathcal{F}(N, K)$. This class includes the well-known family of Reed-Muller (RM) codes, as discussed in detail in [2]. A notable member of this class is the extended Hamming code (EHC), which is a special instance of RM codes. The class can be extended by taking code products in the sense of Elias [3], and includes the products of EHCs, which have been shown by Pyndiah [4] to achieve excellent performance under turbo decoding and are now part of several wireless standards, including the WiMAX standard [5]. The idea of polar coding is to select the best code in the class $\mathcal{F}(N, K)$ for a given channel, and thereby achieve the best performance over all codes in the class $\mathcal{F}(N, K)$. The effectiveness of this idea has been shown in [2] where performance improvements over RM codes with block lengths $N = 256$ were documented. In this paper, we explore implementation architectures for polar codes that can be used for polar coding at significantly higher code block lengths. We illustrate the proposed implementation for a block length $N = 4096$ code, showing that polar codes may be a viable alternative for practical applications.

II. Graph representation of polar codes

The codes in the family $\mathcal{F}(N, K)$ for a fixed N and for all $1 \leq K \leq N$ can be represented using a graph that corresponds to a computational circuit for the transformation $F^{\otimes n}$. For $N = 8$, such a representation is shown in Fig. 1. This circuit computes the transform $x_1^8 = u_1^8 F^{\otimes 3}$ where $u_1^8 = (u_1, \ldots, u_8)$ and $x_1^8 = (x_1, \ldots, x_8)$. In general, we use the notation a_1^N to denote a vector $(a_1, \ldots, a_N)$.

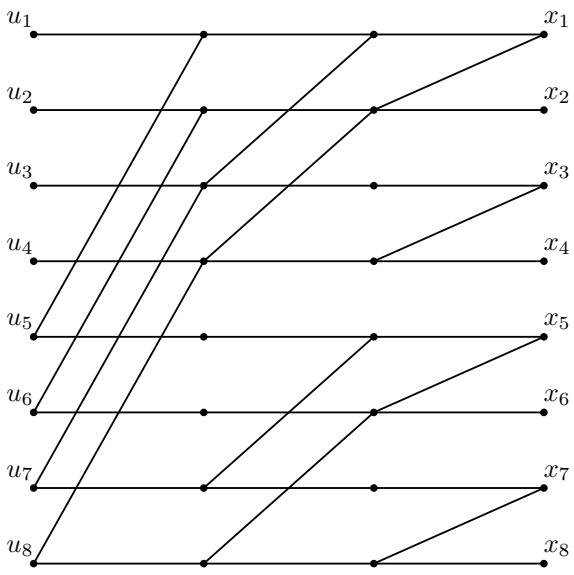

Fig. 1. The transformation $F_2^{\otimes 3}$. Each edge between two nodes carries a value 0 or 1 from the left node to the right node. At each node, all values arriving from the left are added modulo-2 and the result is forwarded on all outgoing edges to the right.

The circuit in Fig. 1 can be used as a universal encoder for all codes in the class $\mathcal{F}(8, K)$, $1 \leq K \leq 8$. For example, for

the (8,4) EHC, we set the inputs u_1, u_2, u_3 and u_5 equal to zero (corresponding to rows of $F^{\otimes 3}$ with Hamming weights 2 or less). The remaining inputs are left free to carry user data bits.

The sparseness of the graph representation for the transformation $F^{\otimes n}$ suggests that Gallager's belief propagation algorithm [6] may be an effective decoding method for codes in the class $\mathcal{F}(N, K)$. Indeed, this point was noticed by Forney [7] who suggested a BP decoder for RM codes using *factor graph* representations. We will follow Forney and use factor graphs here. The factor graph for the family of codes $\mathcal{F}(N, K)$ is given in Fig. 2 for $N = 8$.

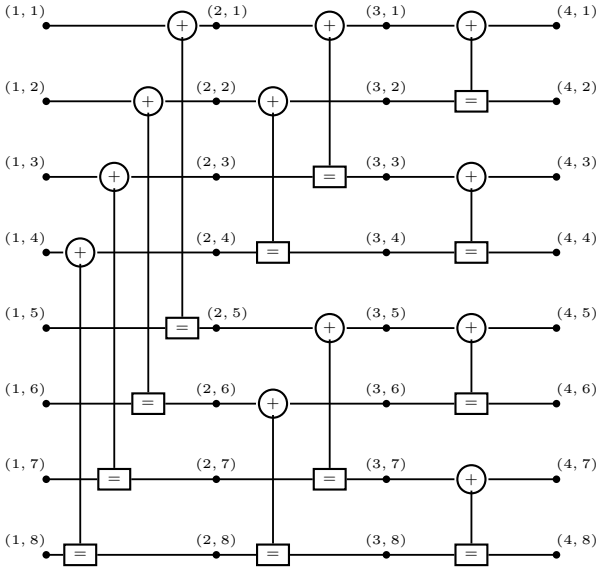

Fig. 2. The factor graph representation for the transformation $F^{\otimes 3}$.

The nodes of the factor graph are labeled with pairs of integers (i, j), $1 \le i \le n+1$, $1 \le j \le N$. From the decoder's perspective, the leftmost nodes, $(1, j)$, are associated with the source data u_j that are to be estimated, while the rightmost nodes $(n+1, j)$ are associated with channel input variables x_j that are observed through a noisy channel. Apart from these source and channel nodes, each node (i, j) in the factor graph has a 0-1 value which the decoder needs to estimate. The BP decoder tackles this task by associating two messages with each node (i, j): a right-propagating message $R_{i,j}^{(t)}$ and a left-propagating message $L_{i,j}^{(t)}$, where $t = 0, 1, \ldots$ is a time index. These message correspond to likelihood ratios at time t and are initialized as

$$L_{n+1,j}^{(0)} = \frac{P(x_j = 0|y_j)}{P(x_j = 1|y_j)}$$

$$R_{n+1,j}^{(0)} = \frac{P(u_j = 0)}{P(u_j = 1)} = \begin{cases} 1 & \text{if } j \text{ is an information coordinate} \\ \infty & \text{if } j \text{ is a frozen coordinate.} \end{cases}$$

All other $R_{i,j}^{(0)}$ and $L_{i,j}^{(0)}$ are set equal to 1. Note that setting $R_{1,j}^{(0)} = \infty$ for j a frozen coordinate makes sense since the decoder knows that such coordinates are set equal to 0 at the encoder. On the other hand, setting $R_{1,j}^{(0)} = 1$ for j an information coordinate indicates that the a-priori 0 and 1 are equally likely values for such coordinates.

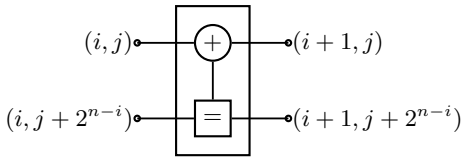

Fig. 3. The basic computational block of BP decoder for a polar code. The node labels indicate the possible position of the block in the decoder factor graph.

The basic computational element of BP decoding is a 4 terminal processing block as shown in Fig. 3. We observe that in each stage of the example of Fig. 2, there are 4 such computational blocks. In a general factor graph for decoding a block length N polar code, there are a total of $\frac{1}{2}N \log N$ such blocks. This computational block implements the mapping

$$\begin{aligned} L_{i,j}^{(t+1)} &= f(L_{i+1,j}^{(t)},\, L_{i+1,j+N_i}^{(t)}\, R_{i,j+N_i}^{(t)}) \\ L_{i,j+N_i}^{(t+1)} &= L_{i+1,j+N_i}^{(t)}\, f(L_{i+1,j}^{(t)},\, R_{i,j}^{(t)}) \\ R_{i+1,j}^{(t+1)} &= f(R_{i,j}^{(t)},\, L_{i+1,j+N_i}^{(t)}\, R_{i+1,j+N_i}^{(t)}) \\ R_{i+1,j+N_i}^{(t+1)} &= R_{i,j+N_i}^{(t)}\, f(R_{i,j}^{(t)},\, L_{i+1,j}^{(t)}) \end{aligned}$$

where $N_i = 2^{n-i}$ and $f(x, y) = (1+xy)/(x+y)$ for any two reals x, y.

III. Uniform encoder-decoder architectures

Although the factor graph representation in Fig. 2 defines efficient encoder and decoder architectures for implementation of encoders and decoders both in software and hardware, the non-uniform structure of the graph from one stage to next hinders re-usability of processing modules. A more uniform architecture is desirable for flexible implementations where circuit complexity can be traded off for time complexity. For this we give alternative architecture given in Fig. 4. Here R is the *reverse-shuffle* operator that transforms an input vector v_1^N of length N for any even integer N into $(v_1, v_3, \ldots, v_{N-1}, v_2, v_4, \ldots, v_N)$, and $\oplus$ is the addition operation that transforms any 0-1 vector v_1^N of even length into $(v_1 \oplus v_2, v_2, v_3 \oplus v_4, v_4, \ldots, v_{N-1} \oplus v_N, v_N)$ where $\oplus$ is modulo 2 addition. We should note that although the circuit in Fig. 4 is end-to-end equivalent to that in Fig. 2 in that they implement the same transformation $F^{\otimes}$, they are not equivalent stage-by-stage.

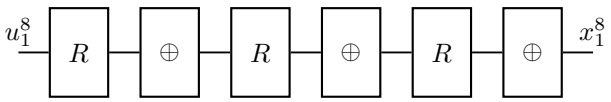

Fig. 4. The uniform factor graph representation for the transformation $F^{\otimes 3}$.

The uniform factor graph representation shown above for the case of block length $N = 8$ holds for N any power of 2.

For $N = 2^n$, one only needs to use n copies of the tandem R and $\oplus$ operations suitable for vectors of length N.

Another uniform factor graph representation of $F^{\otimes 3}$ is shown in Fig. 5, where S is the *shuffle* operator that transforms an input vector v_1^N of even length N into $(v_1, v_{N/2+1}, v_2, v_{N/2+2}, \ldots, v_{N/2}, v_N)$. It is easy to see that this circuit implements the inverse of the transform in circuit Fig. 4. But since the inverse of the transform $F^{\otimes n}$ is itself, the claim follows. There are many other uniform factor graph realizations of the transform $F^{\otimes n}$. The availability of such uniform representations is important for hardware implementations where the same block may be re-used for reduced complexity.

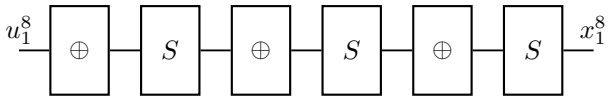

Fig. 5. Another uniform factor graph realization for $F^{\otimes 3}$.

IV. Summary

We have given some implementation architectures for polar codes that are suitable for hardware and software implementations. These architectures are based on a small set of re-usable modules and allow pipelined implementations.

Acknowledgement

This work was supported in part by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under project no 107E216 and in part by the European Community under FP7 Network of Excellence project NEWCOM++ (grant no 216715) and FP7 STREP project WiMAGIC (grant no 215167).

References

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, pp. 3051–3073, July 2009.

[2] E. Arıkan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Comm. Letters*, vol. 12, pp. 447–449, June 2008.

[3] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. PGIT-4, pp. 29–37, September 1954.

[4] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Communications*, vol. 46, pp. 1003–1010, Aug. 1998.

[5] IEEE LAN/MAN Standards Committee, *IEEE Std 802.16-2004 (Revision of IEEE Std 802.16-2001)*. IEEE: New York, NY, 2004.

[6] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[7] G. D. Forney Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. IT-47, pp. 520–548, Feb. 2001.