An FPGA Implementation Architecture for Decoding of Polar Codes

Alptekin Pamuk

Department of Electrical-Electronics Engineering
Bilkent University
Ankara, TR-06800, Turkey
apamuk@bilkent.edu.tr

Abstract **P olar codes are a class of codes versatile enough to achieve the Shannon bound in a large array of source and channel coding problems. For that reason it is important to have ef cient implementation architectures for polar codes in hardware. Motivated by this fact we propose a belief propagation (BP) decoder architecture for an increasingly popular hardware platform; Field Programmable Gate Array (FPGA). The proposed architecture supports any code rate and is quite exible in terms of hardware complexity and throughput. The architecture can also be extended to support multiple block lengths without increasing the hardware complexity a lot. Moreover various schedulers can be adapted into the proposed architecture so that list decoding techniques can be used with a single block. Finally the proposed architecture is compared with a convolutional turbo code (CTC) decoder for WiMAX taken from a Xilinx Product Speci cation and seen that polar codes are superior to CTC codes both in hardware complexity and throughput.**

Index Terms **P olar codes, belief propagation decoding, bp decoder, hardware implementation, FPGA.**

I. Introduction

Polar coding is a type of error-correction coding method that has been introduced recently in [1]. The main motivation for the introduction of polar coding was theoretical, namely, to give an explicit code construction that is *provably* capacity-achieving and implementable with low-complexity. Other code families that are practically implementable and known to have capacity-achieving performance, most notably, turbo and LDPC codes, still lack exact mathematical proofs that they do indeed achieve capacity except for some special cases. In contrast polar codes yield to precise mathematical analysis. Polar coding has also proven to be a versatile coding method capable of achieving the information-theoretic limits in a wide range of source and channel coding problems; for a representative list of such work, we cite [2], [3], [4], [5], [6], [7].

Although polar codes were initially introduced as a theoretical device, it was recognized from the start that they might be useful for practical purposes, too. In terms of the code block-length N, the encoding and decoding complexities of polar coding are approximately $N \log_2 N$ [1]; and, for any rate below channel capacity, the probability of frame error for polar codes goes to zero roughly as $\mathcal{O}(2^{-\sqrt{N}})$ [8], [9], [10], [11]. It is noteworthy that the error probability for polar codes does not suffer from an error - oor effect.

Experimental studies on polar coding for source and channel coding have been reported in [12], [2], [13], [14]. For channel coding applications, polar codes have not yet been shown to perform nearly as good as the current state-of-the-art codes. Improving the performance of polar codes by techniques such as list-decoding or as part of an ARQ (automatic repeat request) scheme are current research areas. For source coding and rate-distortion coding problems, polar codes have been shown to be competitive with the state-of-the-art methods [2], [3].

Whether polar coding will eventually have a practical impact is an open question at this time; the answer may depend on a better understanding of the complexity vs. performance offered by polar codes. Polar codes have a recursive structure that makes low-complexity implementations possible. An initial discussion of implementation options for polar codes were given in [1]. This work was continued in [15] which discussed a more speci c hardware implementation option suitable for pipelining. The present paper extends the ideas in [15] and presents actual hardware implementations on speci c FPGA platforms. Other work on this subject includes [16] which discusses VLSI implementation options for polar codes.

In this paper, we focus on the implementation of a BP decoder for polar codes. We omit discussion of encoder implementations due to space limitations. We also omit the discussion of how to implement a successive cancellation (SC) decoder. These will be the subject of an extended version of this work. The main contribution of the paper is to give an actual hardware implementation of polar codes and provide complexity gures derived from them. We also give a brief comparison of polar codes with CTC used in the 802.16e standard, which indicate clearly an advantage for polar codes. These results show that a more comprehensive comparison of polar codes with CTC and other state-of-the-art codes in terms of complexity and performance is warranted.

The rest of the paper is organized as follows. Section II describes brie y polar encoding and decoding algorithms. Section III-C presents the FPGA implementation architecture for a BP decoder for polar codes. Section IV presents comparisons between polar coding and CTC codes. The paper concludes with some remarks in Section V.

Notation: The codes considered are over the binary eld

978-1-61284-402-2/11/$26.00 ©2011 IEEE

$\mathbb{F}_2$. All vectors and matrices and operations on them are over $\mathbb{F}_2$ unless otherwise indicated. We use boldface uppercase(lowercase) are used to denote matrices(vectors). $\mathbf{I}_n$ stands for $n \times n$ identity matrix for any $n \geq 1$. For any two matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \otimes \mathbf{B}$ denotes their Kronecker product, $\mathbf{A}^{\otimes n}$ denotes the nth Kronecker power of $\mathbf{A}$.

II. Polar codes

Polar coding a linear block coding method where the code block-length N can be any power of two and the code rate can be adjusted to any number K/N with $0 \leq K \leq N$. For a given $n \geq 1$, a polar code with block-length $N = 2^n$ and rate K/N is obtained by the linear mapping

$$\mathbf{x} = \mathbf{u}\mathbf{G}_N, \quad \mathbf{G}_N = \mathbf{F}^{\otimes n}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{1}$$

where $\mathbf{u}$ and $\mathbf{x}$ are row vectors of length N and $\mathbf{G}_N$ is an N-by-N matrix, all over $\mathbb{F}_2$. We regard $\mathbf{u}$ as the data vector and $\mathbf{x}$ as the codeword that gets sent over a binary-input channel. For a rate K/N polar code, $N - K$ of the coordinates of $\mathbf{u}$ need to be frozen and the remaining K are to be left free (to take on any one of the 2^K possible values). Thus, a polar code of rate K/N is speci ed by specifying a K-element subset $\mathcal{A} \subset \{1, \ldots, N\}$ which designates the free coordinates. Given such a subset $\mathcal{A}$, the subvector $\mathbf{u}_\mathcal{A} \stackrel{\Delta}{=} (u_i : i \in \mathcal{A})$ carries the user data that is free to change in each round of transmission, while the complementary subvector $\mathbf{u}_{\mathcal{A}^c} \stackrel{\Delta}{=} (u_i : i \in \mathcal{A}^c)$, where $\mathcal{A}^c$ denotes complement of $\mathcal{A}$ in $\{1, \ldots, N\}$, stays x ed throughout the session and is known to the decoder. In [1] a method is described for determining the set $\mathcal{A}$ for a given channel. For the purposes of this paper, the way $\mathcal{A}$ is computed is not important. The implementations described take $\mathcal{A}$ as a given parameter.

Decoders for polar codes can be implemented by using factor graph representations of the equation (1) as described in [1], [12]. There exist many factor graph representations that are simple permutations of each other. Some representations that are especially suitable for reuse of hardware modules in a pipelined architecture were described in [15]. To be more precise, we seek representations of the generator matrix in the form $\mathbf{G}_N = \mathbf{M}^n$ which allows reuse of the module represented by $\mathbf{M}$. We have found six such choices for $\mathbf{M}$, namely,

$$\begin{aligned} \mathbf{M}_1 &= \mathbf{S}(\mathbf{I}_{N/2} \otimes \mathbf{F}), & \mathbf{M}_2 &= (\mathbf{I}_{N/2} \otimes \mathbf{F})\mathbf{S}, \\ \mathbf{M}_3 &= \mathbf{S}(\mathbf{F} \otimes \mathbf{I}_{N/2}), & \mathbf{M}_4 &= (\mathbf{I}_{N/2} \otimes \mathbf{F})\mathbf{S}^T, \\ \mathbf{M}_5 &= \mathbf{S}^T(\mathbf{I}_{N/2} \otimes \mathbf{F}), & \mathbf{M}_6 &= (\mathbf{F} \otimes \mathbf{I}_{N/2})\mathbf{S}^T. \end{aligned}$$

In these equations $\mathbf{S}$ and $\mathbf{S}^T$ correspond to the *shuf e* and *reverse-shuf e* permutation operations as described in [15]. In the remainder of the paper we consider implementations based on $\mathbf{M} = \mathbf{M}_1$.

We will describe the factor graph representations of $\mathbf{G}_N = \mathbf{M}_1^n$ with reference to the small example shown in Fig. 1 for $n = 3$. The factor contains $N(n+1)$ nodes with each node labeled with a pair of integers (i, j), $1 \leq i \leq n$, $1 \leq j \leq N/2$. The rst element of (i, j) designates the *layer* and the second element the index of the node within that level. The nodes at layer 1 are associated with the source vector $\mathbf{u}$, the nodes at layer $(n+1)$ are associated with the codeword $\mathbf{x}$. Nodes in the factor graph appear in groups of four, as forming the *ports* of 2-by-2 *basic computational blocks* (BCB) as shown in Fig. 2.

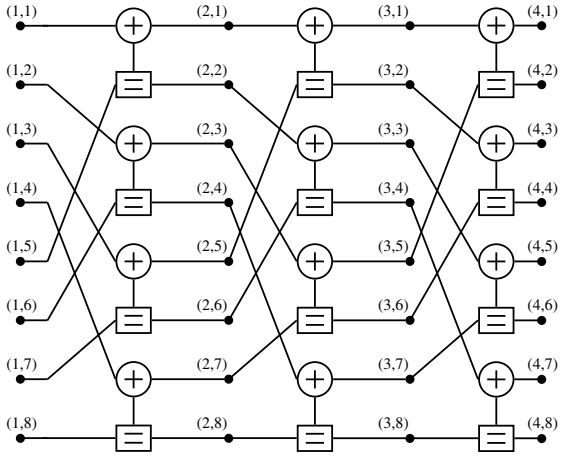

Fig. 1. Uniform factor graph representation for $\mathbf{G}_8$.

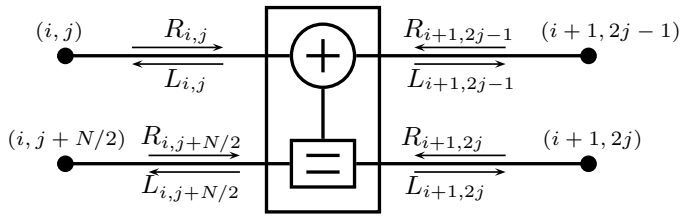

Fig. 2. Basic Computational block of BP decoder. The node labels are assigned according to the uniform factor graph representation which is depicted in Fig. 1.

The decoders we consider are message-passing decoders where we assign BCBs the task of computing the messages and the nodes simply relay the messages between neighboring BCBs. A message that crosses a node (i, j) from right to left (left to right) is designated by $L_{i,j}$ ($R_{i,j}$) as depicted in Fig. 2. The messages represent the log-likelihood ratios (LLRs) and are computed as (see [1] and [15] for details) using the formulas

$$\begin{aligned} L_{i,j} &= g(L_{i+1,2j-1}, L_{i+1,2j} + R_{i,j+N/2}) \\ L_{i,j+N/2} &= g(R_{i,j}, L_{i+1,2j-1}) + L_{i+1,2j} \\ R_{i+1,2j-1} &= g(R_{i,j}, L_{i+1,2j} + R_{i,j+N/2}) \\ R_{i+1,2j} &= g(R_{i,j}, L_{i+1,2j-1}) + R_{i,j+N/2} \end{aligned} \tag{2}$$

where $g(x, y) = \ln((1 + xy)/(x + y))$. We approximated this function by $g(x, y) \approx \operatorname{sign}(x)\operatorname{sign}(y)\min(|x|, |y|)$ in implementations. The messages $R_{1,j}$, $1 \leq j \leq N$, that

emanate from the source are x ed throughout as

$$R_{1,j} = \begin{cases} 0 & \text{if } j \in \mathcal{A} \\ \infty & \text{if } j \in \mathcal{A}^c \text{ and } u_j = 0 \\ -\infty & \text{if } j \in \mathcal{A}^c \text{ and } u_j = 1 \end{cases}$$

The messages $L_{n+1,j}$, $1 \leq j \leq N$, that originate from the channel block are given by

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

where $P(y_j|x_j)$ denotes the probability that the channel output y_j is received when codeword element x_j is sent.

III. FPGA Implementation

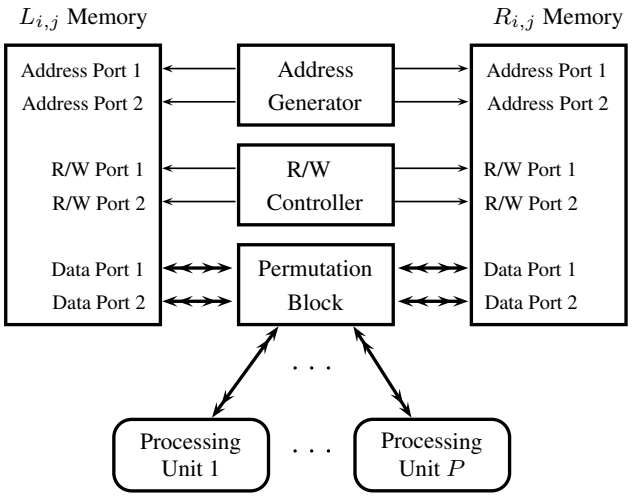

Fig. 3. A top level block diagram of the proposed BP decoder architecture.

The main blocks of the proposed decoder architecture are shown in Fig. 3. Introductory information about the blocks will be given rst.

- Processing Units (PU) are the functional blocks of the decoder that implement the mathematical expressions given in (2). The number of PU's are denoted by P which is assumed to be a power of 2 throughout the paper. One can make a compromise between the complexity and throughput by playing with P. The PU's are pipelined blocks and their latency in clock cycles (CC) is denoted by D.
- There are two independent and dual-port memory blocks each of which stores the left or right propagating messages at all layers.
- Address Generator (AG) controls the scheduling of the decoder.
- R/W Controller (RWC) controls the data o w between the memories and the BCB's, because in each memory access either a read or a write operation is performed.
- Permutation Block (PB) adapts the order of messages in the memory cells to the PU's inputs and outputs.

A. Decoder Parallelization by Message Grouping

Considering the factor graph representation in Fig. 1 there are $(n+1)N$ left propagating (LP) and $(n+1)N$ right propagating (RP) messages in the decoder. If there is only one PU, then one iteration will require at least nN CC's which is not practical. Therefore we propose a method such that more than one PU's can work in parallel without any memory con icts.

First divide N LP (RP) messages at a layer into N/P sets each of which consists of P LP (RP) messages. Let $\mathcal{L}^{(i,k)}$ ($\mathcal{R}^{(i,k)}$) denote the set for LP (RP) messages where i is the layer index and k is the set index within that layer ($1 \leq k \leq N/P$). Also let the elements of $\mathcal{L}^{(i,k)}$ and $\mathcal{R}^{(i,k)}$ to be

$$\begin{aligned} \mathcal{L}^{(i,k)} &= \{L_{i,(k-1)P+j} \mid 1 \leq j \leq P\} \\ \mathcal{R}^{(i,k)} &= \{R_{i,(k-1)P+j} \mid 1 \leq j \leq P\} \end{aligned} \tag{3}$$

With this background it can be easily seen that the elements of the sets $\mathcal{R}^{(i,k)}$, $\mathcal{R}^{(i,k+N/(2P))}$, $\mathcal{L}^{(i+1,2k-1)}$ and $\mathcal{L}^{(i+1,2k)}$ form the inputs of a group of P BCB's. Likewise the elements of the sets $\mathcal{L}^{(i,k)}$, $\mathcal{L}^{(i,k+N/(2P))}$, $\mathcal{R}^{(i+1,2k-1)}$ and $\mathcal{R}^{(i+1,2k)}$ form the outputs of the same group. As a result if a memory cell contains a set, then P PU's can work in parallel without any memory con icts.

Note that this method does not put any restriction on the storage addresses of the sets. We suggest to store the set $\mathcal{L}^{(i,k)}$/$\mathcal{R}^{(i,k)}$ at the memory address $(i-1)N/P + (k-1)$ because this provides simpli cation in the hardware.

B. Decoding Process

We showed a sample decoding process in Fig. 4 to form a basis for the general decoding procedure. The column labels R and W indicate whether the messages in that column are read from the memory or written to the memory. The row label PU-1(PU-2) indicates that the messages in this row are related with the rst PU(second PU). The outputs appear four CC's after their corresponding inputs because D is assumed to be 4 in this example. At the 11^{th} and 12^{th} CC's there are no R/W operations, because there is no message left to be processed for that iteration. However these idle CC's will be lled if the decoding continues with the next iteration. It is worth to note here that we obtained good performance by processing the trellis from right to left in the odd iterations (as in Fig. 4) and from left to right in the even iterations.

In general 4sets/CC are read from the memories until the rst outputs are ready at the PU's. This is called the memory read cycle and continues D CC's. When the rst outputs are ready, memory write cycle begins. In this cycle 4sets/CC are written to memories and this cycle is completed again in D CC's. As a result $4PD$ messages are computed in $2D$ CC's. There are a total of $2Nn$ messages to be computed for each iteration. Hence one iteration is completed in Nn/P CC's.

Memory read and memory write cycles are controlled by RWC whose time utilization is drawn in Fig. 5.

CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R/W	R	R	R	R	W	W	W	W	R	R			W	W
PU-1	$R_{3,1}$	$R_{3,3}$	$R_{2,1}$	$R_{2,3}$	$L_{3,1}$	$L_{3,3}$	$L_{2,1}$	$L_{2,3}$	$R_{1,1}$	$R_{1,3}$			$L_{1,1}$	$L_{1,3}$
	$R_{3,5}$	$R_{3,7}$	$R_{2,5}$	$R_{2,7}$	$L_{3,5}$	$L_{3,7}$	$L_{2,5}$	$L_{2,7}$	$R_{1,5}$	$R_{1,7}$			$L_{1,5}$	$L_{1,7}$
	$L_{4,1}$	$L_{4,5}$	$L_{3,1}$	$L_{3,5}$	$R_{4,1}$	$R_{4,5}$	$R_{3,1}$	$R_{3,5}$	$L_{2,1}$	$L_{2,5}$			$R_{2,1}$	$R_{2,5}$
	$L_{4,2}$	$L_{4,6}$	$L_{3,2}$	$L_{3,6}$	$R_{4,2}$	$R_{4,6}$	$R_{3,2}$	$R_{3,6}$	$L_{2,2}$	$L_{2,6}$			$R_{2,2}$	$R_{2,6}$
PU-2	$R_{3,2}$	$R_{3,4}$	$R_{2,2}$	$R_{2,4}$	$L_{3,2}$	$L_{3,4}$	$L_{2,2}$	$L_{2,4}$	$R_{1,2}$	$R_{1,4}$			$L_{1,2}$	$L_{1,4}$
	$R_{3,6}$	$R_{3,8}$	$R_{2,6}$	$R_{2,8}$	$L_{3,6}$	$L_{3,8}$	$L_{2,6}$	$L_{2,8}$	$R_{1,6}$	$R_{1,8}$			$L_{1,6}$	$L_{1,8}$
	$L_{4,3}$	$L_{4,7}$	$L_{3,3}$	$L_{3,7}$	$R_{4,3}$	$R_{4,7}$	$R_{3,3}$	$R_{3,7}$	$L_{2,3}$	$L_{2,7}$			$R_{2,3}$	$R_{2,7}$
	$L_{4,4}$	$L_{4,8}$	$L_{3,4}$	$L_{3,8}$	$R_{4,4}$	$R_{4,8}$	$R_{3,4}$	$R_{3,8}$	$L_{2,4}$	$L_{2,8}$			$R_{2,4}$	$R_{2,8}$

Fig. 4. A sample decoding process is shown for $N = 8$, $P = 2$ and $D = 4$. The column labels R and W indicate whether the messages in that column are read from the memory or written to the memory. The row label PU-1(PU-2) indicates that the messages in this row are related to the rs t PU(second PU).

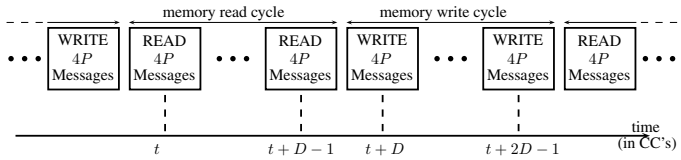

Fig. 5. Time utilization of RWC. Memory read and memory write cycles follow each other till the end of decoding.

C. Salient Features of the Proposed Decoder Architecture

The salient features of the decoder can be listed as:

- Flexibility: By changing the number of PU's the designer can make compromise between the hardware complexity and throughput.
- Multiple code rate support: The code rate can be changed at the run time and this feature does not bring any additional hardware complexity.
- Multiple code length support: The codewords whose length is a power of 2 and smaller than N can also be decoded with the same decoder by adding an extra AG for each code length. The hardware cost of adding an extra AG is very small compared to the overall complexity.
- Multiple scheduler support: It is known that the performance of BP decoding heavily depends on the scheduling [3]. We did not put any restriction on the scheduling once the sets are selected appropriately. Therefore the proposed architecture has the e xibility to support various schedulers by simply adding an extra AG for each scheduling. This feature enables list decoding with a single hardware.

D. Synthesis Results

The main resources of our FPGA's will introduced before the synthesis results.

- *Look-up Table (LUT):* The function generator of our FPGA which can implement any function with 4-binary-inputs and 1-binary-output or 6-binary-inputs and 1-binary-output depending on the FPGA.
- *Flip- op (FF):* FF is the well known synchronous D-type registers with 'set/reset' inputs.
- *Block-RAM (BRAM):* The dedicated memory blocks of the FPGA. Their size and number of ports may vary depending on the FPGA.

The hardware complexity of the decoder heavily depends on the number of PU's. There are also multiplexers inside the PB whose complexities are proportional to P. Therefore LUT and FF usage is proportional to P.

The width of BRAM cells are 32 or 64 bits at maximum depending on the the FPGA. Therefore if $(P \times$ message size in bits$)$ is larger than the width of a single BRAM, then the synthesis tool concatenates multiple BRAM's to produce a memory with larger cell width. Likewise if $(n + 1)N/P$ is larger than the number of cells of a single BRAM, then the synthesis tool cascades multiple BRAM's to implement this memory. In conclusion P and N both affect the number of utilized BRAM's, however as we will see from the synthesis results N is more effective.

The proposed decoder is implemented on a Xilinx Virtex-4 FPGA. Messages are represented by 6 bits. The synthesis results are listed in Table I[1]. In the rst part of the table P is kept constant and N is increased. In the second part N is kept constant and P is increased.

TABLE I
RESOURCE UTILIZATION FOR VARIOUS BLOCK SIZES: XC4VSX25

Block Size	P	LUT	FF	BRAM
256	16	2779	1592	6
512	16	2809	1596	6
1024	16	2794	1600	12
2048	16	2797	1604	22
4096	16	2805	1605	48
8192	16	2808	1612	96
2048	1	300	179	24
2048	2	462	271	24
2048	4	792	459	24
2048	8	1459	839	24
2048	16	2797	1605	22
2048	32	5479	3144	22

IV. COMPARISONS WITH CTC

In this section, we compare FPGA implementation of our polar decoder with a CTC decoder for WiMAX which is taken from a Xilinx Product Speci cation for CTC Decoders [17]. The CTC decoder supports all the CTC con gurations in WiMAX whose block sizes are smaller than 960 bits. Therefore its hardware complexity is independent from block size and code rate. The polar decoder used in the comparisons has 16 PU's and supports a single block size with any coding rate. Both decoders use 6 bit messages.

Table II compares the throughput attainable by the two decoders, where each decoder is set to operate at 5 iterations. The FPGA clock is set to 160 MHz. We see a clear advantage for polar codes.

Table III compares the resource utilization for the above two decoders for the Xilinx XC5VLX85 FPGA. The resource utilization for the polar code is much lower.

The above results show that the BP decoder for polar coding has a remarkably lower complexity and high throughput compared to the CTC decoder. This combined with the universal

[1]The synthesis is done using Xilinx-ISE version 12.2.

TABLE II
THROUGHPUT COMPARISON: 160 MHZ, 5 ITERATIONS.

Code	Data rate (Mbps)
CTC (960,480)	30.59
Polar (1024,512)	27.83
CTC (576,480)	30.59
Polar (512,426)	52.03
CTC (288,144)	27.73
Polar (256,128)	35.56
CTC (192,144)	27.73
Polar (256,192)	53.33

TABLE III
RESOURCE UTILIZATION: XC5VLX85

Code	LUT	FF	BRAM	DSP48
CTC (960,480)	6611	6767	9	4
Polar (1024,512)	2324	1502	6	0
CTC (576,480)	6611	6767	9	4
Polar (512,426)	2316	1498	6	0
CTC (288,144)	6611	6767	9	4
Polar (256,128)	2309	1494	6	0
CTC (192,144)	6611	6767	9	4
Polar (256,192)	2309	1494	6	0

nature of the polar coding scheme offers distinct advantages for polar codes.

On the other hand performance of CTC codes is better than polar codes as seen in Fig. 6. Even though maximum iteration count is set to 5 for CTC and 50 for polar code in the simulations, there is 1-2dB gap between their performances at 1E-3 PER. There are studies to close this gap, however their hardware complexities should also be studied.

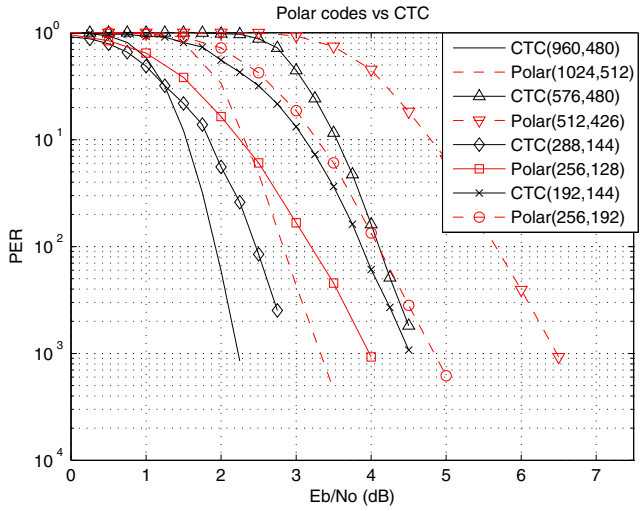

Fig. 6. PER comparison of polar codes under BP decoding and CTC codes.

V. CONCLUSION

In this paper we proposed a belief propagation decoder architecture which is optimized for FPGA's. The proposed architecture is quite e xible in the sense of hardware utilization and can support multiple coding rates without any additional cost. The decoder can also be extended to support multiple code lengths and different schedulers with a little extra hardware utilization.

The FPGA utilization of the proposed decoder is compared with a CTC decoder taken from a Xilinx Product Speci cation. It is shown that polar codes are superior to CTC codes both in hardware complexity and throughput. Even though PER performance of the polar codes under BP decoding is a few dB's behind the CTC codes, polar coding is still a promising technique because of aforementioned advantages.

ACKNOWLEDGMENT

This work was supported in part by The Scienti c and Technological Research Council of Turkey (TUBITAK) under contract no. 110E243.

REFERENCES

[1] E. Ar kan, 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] N. Hussami, S. B. Korada, and R. Urbanke, Performance of polar codes for channel and source coding, in *Proc. 2009 IEEE Int. Symp. Inform. Theory*, (Seoul, South Korea), pp. 1488 1492, 28 June - 3 July 2009.
[3] S. B. Korada, *Polar codes for channel and source coding*. PhD thesis, EPFL, Lausanne, 2009.
[4] E. Sasoglu, E. Telatar and E. Yeh, Polar codes for the two-user multiple-access channel, Proceedings of the 2010 IEEE Information Theory Workshop, Cairo, Egypt, January 6-8, 2010.
[5] E. Abbe and E. Telatar, MA C polar codes and matroids, Information Theory and Applications Workshop, ITA 2010 , pp.1-8, Jan. 31 - Feb. 5, 2010.
[6] E. Hof, I. Sason and S. Shamai, Polar coding for reliable communications over parallel channels, Proceedings of the 2010 IEEE Information Theory Workshop, Dublin, Ireland, Aug. 30 - Sept. 3, 2010.
[7] H. Mahdavifar and A. Vardy, Achieving the secrecy capacity of wiretap channels using polar codes, to appear in the *IEEE Trans. Inform. Theory,* vol. 57, 2011. (Also available on arXiv:1007.3568.)
[8] E. Ar kan and E. Telatar, On the rate of channel polarization, 2009 IEEE Int. Symp. Inform. Theory, Seoul, Korea, 28 June - 3 July 2009.
[9] S. H. Hassani, R. Urbanke, On the scaling of polar codes: 1. The behavior of polarized channels, 15 Jan 2010, arXiv:1001.2766v2 [cs.IT]
[10] T. Tanaka and R. Mori, Re ned rate of channel polarization, 13 Jan 2010, arXiv:1001.2067v1 [cs.IT]
[11] T. Tanaka, On speed of channel polarization, *Proc. 201 IEEE Inform. Theory Workshop (ITW 2010 Dublin),* Dublin, Ireland, Aug. 30 - Sept. 3, 2010.
[12] E. Ar kan, A performance comparison of polar codes and Reed-Muller codes, *IEEE Comm. Lettrs.*, vol. 12, no. 6, pp. 447-449, June 2008.
[13] E. Ar kan and H. Kim and G. Markarian and U. Ozgur and E. Poyraz, Performance of short polar codes under ML decoding, ICT Mobile Summit, Santander, Spain, June 2009.
[14] E. Ar kan and G. Markarian, T wo-dimensional polar coding, International Symposium on Coding Theory and Applications (ISCTA 2009), Ambleside, UK, Jul. 2009.
[15] E. Ar kan, Polar codes: A pipelined implementation, *Proc. 4th Int. Symp. Broadband Communications (ISBC2010)*, Melaka, Malaysia, 11-14 July 2010.
[16] C. Leroux, I. Tal, A. Vardy, W. J. Gross, Hardw are architectures for successive cancellation decoding of polar codes, The 36th International Conference on Acoustics, Speech and Signal Processing (ICASSP, 2011), Prague, May 22-27, 2011.
[17] XILINX, IEEE 802.16e CTC Decoder v4.0, 2009.