


# Computational Complexity Reduced Belief Propagation Algorithm for Polar Code Decoders

Arvid B. van den Brink \*  
Email: a.b.vandenbrink@utwente.nl

Marco J.G. Bekooij\*<sup>†</sup>  
Email: marco.bekooij@nxp.com

\*Department of Computer Architectures  
for Embedded Systems  
University of Twente, Enschede, The Netherlands

<sup>†</sup>Department of Embedded Software  
and Signal Processing  
NXP Semiconductors, Eindhoven, The Netherlands

**Abstract**—The belief propagation algorithm is desirable for a polar code based decoder, because of the potentially low latency and the ability of integration in digital signal processing units or other multi-core processor systems to parallelize the computations. Although belief propagation polar code decoder algorithms have the ability for a highly parallelized implementation, the algorithms require many iterations to achieve a comparable frame error rate and bit error rate with respect to a successive cancellation polar code algorithm. The iterative nature of the belief propagation algorithms also result in a higher computational complexity, i.e.  $\mathcal{O}(IN(2 \log_2 N - 1))$  compared to the computational complexity  $\mathcal{O}(N \log_2 N)$  of the successive cancellation decoder algorithm.

In this paper we propose several simplifications for a simplified belief propagation algorithm for polar code decoders, where the arithmetic complexity of the nodes is reduced.

The proposed belief propagation algorithm shows preliminary results of a net reduction of the arithmetic complexity of  $\approx 13\%$ . This reduction is a result of the reduced number of arithmetic operations, i.e., additions, compares, and multiplications, without a lost in error-correcting performance.

**Index Terms**—Complexity, Belief Propagation, Polar Code, Algorithm

## I. INTRODUCTION

With the introduction of polar codes [1], much attention has been given due to its provably capacity-achieving error correction capability for any binary-input discrete memoryless channel (B-DMC). The low complex decoding algorithm successive cancellation (SC) was proposed for decoding polar codes with a computational complexity  $\mathcal{O}(N \log N)$ , where  $N$  is the length of the codeword. Considering that, if the block length  $N$  increases, the decoding latency and computational complexity emerge as a bottleneck, thus the use of shorter block length is preferred, especially for latency critical applications, e.g. vehicle-to-X communication (V2X), or resource constrained applications such as Internet of Things (IoT) devices.

Recently, multiple possible solutions have been proposed to improve both throughput and performance. Using the successive cancellation list (SCL) or crc-aided successive cancellation list (CA-SCL) [2] algorithms the performance can be enhanced. Alternatively the usage of a larger code length will increase the performance. Nonetheless, for the polar code

to compete, e.g. with the low-density parity check (LDPC) error-correcting code, used in many standards, a polar code with a block length of approximately 16 time larger is required. An obvious disadvantage of this solution is the inherently decreased decoding throughput.

By all of the solutions that improve the throughput, the simplified successive cancellation (SSC) [3] and fast successive cancellation (FSC) [4] algorithms currently deliver the highest improvement in throughput over the original SC algorithm. Polar codes of length  $N$  are formed from two constituent polar codes of length  $N/2$  and using this recursive nature, the decoding of the constituent polar codes directly, if possible and without recursion, will increase the throughput of the decoder.

In this work we focus on the potential improvements of the arithmetic complexity of the polar code BP decoding algorithm. By using simplified equations in the BP algorithm, a new algorithm is obtained with a lower arithmetic complexity, i.e. with less operations like additions and compares. We present a polar code decoder algorithm that, for a  $\mathcal{P}(1024, 512)$  polar code, compared to the textbook BP algorithm, can use approximately 77%, 68%, and 84% of the number of comparisons, additions and multiplications, respectively. Preliminary results, using an ARM Cortex-R4, indicates a net gain of  $\approx 13\%$  in arithmetic complexity. Additionally, the FER and BER of the proposed algorithm is not changed with respect to the BP algorithm.

The remainder of this paper is organized as follows. Section II gives an overview of related work. In Section III we give a review of polar codes in general, and the BP decoding algorithm in particular. Our proposed computational complexity reduced BP decoding algorithm is presented in Section V. In Section VI we describe the methodology used and simulation results. Finally, in Section VII we state the conclusions.

## II. RELATED WORK

Since polar codes are introduced as a capacity achieving code for any binary-input discrete memoryless channel (B-DMC) [1], this type of codes has raised many interest in the research community [5], [2], [6]. In spite of the proven capacity achieving property for infinite code length SC decoding,

the error correcting performance for finite length polar codes is not assured. The natively sequential decoding structure of the SC decoding algorithm is another hurdle to take. This structure results in a lower throughput for longer code lengths. To reduce the error correcting performance degradation of the successive cancellation algorithm, the successive cancellation list decoding algorithm [2] was proposed. The successive cancellation list algorithm explores multiple paths of the decoding tree and chooses the best candidate as the decoder output. Choosing the best candidate will increase the error correcting performance of the decoder, but consequently also increases the computational complexity of the decoder.

To increase the throughput of a polar code decoder, a lot of research is done on the BP algorithm [7], which allows a highly parallel execution. To improve the error correcting performance, a list variant of the BP decoder was proposed by [8]. Reducing the number of iterations, needed for the BP decoding algorithm, several early stopping criteria are proposed in [9], [10], [11]. For increasing the throughput of the BP decoder, several architectural optimizations are proposed in [12]. Reducing the memory requirement of the BP decoder is proposed in [13]. The number of memory operations are reduced by a stage combined BP decoder as proposed in [14]. A vectorized BP decoder is proposed in [15], to address the memory bottleneck of the BP decoder.

### III. PRELIMINARY

#### A. Polar codes

Polar codes are designed as a linear, and capacity achieving block code on any symmetric binary-input discrete memoryless channel (B-DMC), like the binary symmetric channel (BSC). We assume a polar code, as defined in [1], to be represented by a parameter vector  $(N, K, \mathcal{A}, u_{\mathcal{A}^c})$ , with  $N = 2^n \forall n > 1$ , where  $N$  is the length of the code,  $K$  the number of information bits,  $u_{\mathcal{A}^c}$  represents the set of frozen bits, and  $\mathcal{A}$  the set of information bits. The encoding and decoding process of a polar code consists of the following steps:

- 1) The source vector  $u_1^N = (u_1, \dots, u_N)$  is encoded using:

$$x_1^N = u_1^N G_N = u_1^N B_N F^{\otimes n}, \quad (1)$$

where  $G_N$  is the generator matrix,  $B_N$  is the permutation matrix, also known as the bit-reversal matrix, and  $F^{\otimes n}$  is the *Kronecker* power of  $F$  defined as:

$$F^{\otimes n} = F \otimes F^{\otimes(n-1)}, \quad (2)$$

where  $n = \log_2 N$ , and  $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ .

- 2)  $x_1^N$  is sent using the polar code constructed virtual channels  $W^N$ , where the channel output  $y_1^N = (y_1, y_2, \dots, y_N)$  is fed into the decoder.
- 3) Using the knowledge  $\mathcal{A}, u_{\mathcal{A}^c}$  of the polar code and the given channel output  $y_1^N$  the decoder estimates  $\hat{u}_1^N$ .

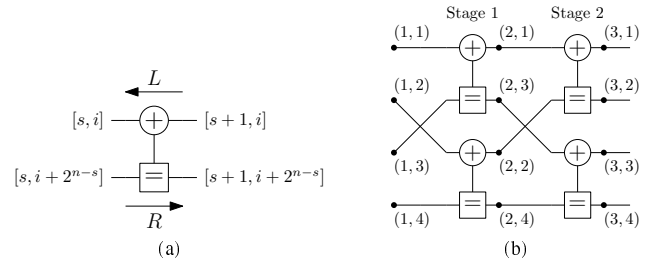


Fig. 1: (a) Belief propagation (BP) computational unit (CU); (b) Polar code factor graph for code length  $N = 4$ .

#### B. Belief propagation polar decoding

Based on the BP polar code algorithm, as described in [7], a belief propagation polar code decoder is created using a factor graph. Fig. 1b shows an example of the created factor graph for a 4-bits BP polar code decoder. Each stage constructed with  $N/2$  computational units (CUs), as shown in Fig. 1a. Each of these CUs consist out of four terminals, as shown in Fig. 1a. These terminals maps to:

$$R_{s+1,i}^{(m+1)} = f(R_{s,i}^{(m)}, L_{s+1,i+2^{n-s}}^{(m)} + R_{s,i+2^{n-s}}^{(m)}), \quad (3)$$

$$R_{s+1,i+2^{n-s}}^{(m+1)} = f(R_{s,i}^{(m)}, L_{s+1,i}^{(m)}) + R_{s,i+2^{n-s}}^{(m)}, \quad (4)$$

$$L_{s,i}^{(m+1)} = f(L_{s+1,i}^{(m)}, L_{s+1,i+2^{n-s}}^{(m)} + R_{s,i+2^{n-s}}^{(m)}), \quad (5)$$

$$L_{s,i+2^{n-s}}^{(m+1)} = f(R_{s,i}^{(m)}, L_{s+1,i}^{(m)}) + L_{s+1,i+2^{n-s}}^{(m)}, \quad (6)$$

with  $1 \leq m \leq M$ , where  $M$  is the maximum number of iterations, and the  $f$  function is calculated according to the min-sum approximation, defined as:

$$f(x, y) \approx 0.9 \cdot \text{sign}(x) \cdot \text{sign}(y) \cdot \min(|x|, |y|). \quad (7)$$

The BP decoding algorithm is an iterative, message passing process using the constructed factor graph. Two types of messages are involved in the decoding process: left bound messages  $L$  and right bound messages  $R$ . Each message, represented by a log likelihood ratio (LLR), is initially assigned an LLR depending on the position at which the node is located in the factor graph.

The left most  $R$  messages are assigned a value according to (8).

$$R_{1,i} = \begin{cases} 0, & \text{if } j \in \mathcal{A}; \\ \infty, & \text{if } j \in \mathcal{A}^c, \end{cases} \quad (8)$$

where  $\mathcal{A}^c$  is the set of frozen bits and  $\mathcal{A}$  is the set of information bits.

The right most  $L$  messages are assigned a value according to the channels LLR using (9).

$$L_{n+1,i} = \ln \left( \frac{\Pr[y_i | x_i = 0]}{\Pr[y_i | x_i = 1]} \right). \quad (9)$$

All other, intermediate node messages in the factor graph are initially set to zero. After  $M$  iterations,  $\hat{u}_1^N$  can be decided

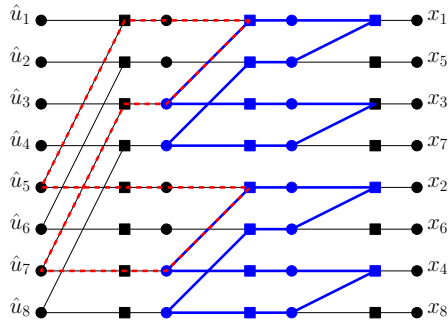


Fig. 2: Several cycles of size 12 in a  $N = 8$  polar code Tanner graph, indicating the girth of this graph. Variable nodes are shown as circles, check nodes as squares.

using threshold detection at the left most terminals using (10).

$$\hat{u}_i = \begin{cases} 0, & \text{if } L_{1,i} + R_{1,i} \geq 0; \\ 1, & \text{otherwise.} \end{cases} \quad (10)$$

### C. Girth of a graph

The girth is the length of the shortest cycle in a graph. In an iterative decoding algorithm, like BP, every variable nodes information remains uncorrelated upto the point the iteration reaches the graphs girth. The higher the girth of a code is, the better the decoding performance the code has, using the BP algorithm. In [16] (and references therein) a technique is proposed to guarantee girth equal or larger then 12, as such a girth is considered more desirable for many different code constructions, like LDPC or polar codes.

In Fig.2 some of the shortest cycles in a polar code with  $N = 8$  are shown. The shoressts cycles in this graph are of length 12. For polar codes with larger code length will also have a girth of 12, due to the recursive nature of the code construction.

## IV. BASIC IDEA

Output values of some nodes in the polar code factor graph are fixed [17], depending on the input values. In [17] only four specific node configurations on the left side of the factor graph are used, in order to reduce the cycle-12 count in the factor graph.

In our proposal we extend this principal to all nodes in the factor graph. For every possible valid combination of input values, the expected output equations are determined. Depending on the input value combinations, the arithmetic complexity of the used equation is equal or less then the currently used BP equations (3)-(6).

## V. PROPOSED ENHANCED BELIEF PROPAGATION ALGORITHM FOR POLAR CODES

For our proposal we observe the inputs and outputs of a single computational unit (CU) as used in the BP polar code decoding algorithm. To simplify the expressions, we denote the left two nodes in Fig. 3 as  $a$  and  $b$  and the right two nodes as  $c$  and  $d$ . The LLRs of any of these nodes will have a value

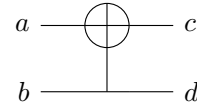


Fig. 3: Simplified belief propagation computational unit.

in  $\mathbb{R}$ , but there are three specific values of interest, i.e. 0,  $+\infty$ , and  $-\infty$ . Since  $+\infty$ , and  $-\infty$  are very hard to implement, we will assume that a maximum number given for the number representation is selected, which replaces  $+\infty$ , and  $-\infty$  with  $+\mathbb{U}max$  and  $-\mathbb{U}max$  respectively.

Depending on the input values, the equations (3)-(6) for the associated output can be simplified into several other equations, as we will explain in Sec. V-A to Sec. V-D.

### A. Simplifications for the right bound upper node

Like the definition in (3), the equation mapped to the upper right node  $c$  in Fig. 3 in simplified form is as follows:

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b + L_d) \cdot \min(|R_a|, |R_b + L_d|) \quad (11)$$

With  $\mathbb{U}max$  sufficiently large, it is save to assume that if  $R_b = \pm\mathbb{U}max$ , then  $R_b + L_d \approx R_b$ .

By an exhaustive search of all possible special input combinations in (11), including the case when no node has a special value, we get the following list of simplified equations:

$$1) R_a = 0 \Rightarrow$$

$$R_c \approx s \cdot \text{sign}(0) \cdot \text{sign}(R_b + L_d) \cdot \min(0, |R_b + L_d|) \\ \approx s \cdot \text{sign}(R_b + L_d) \cdot 0 = 0 \quad (12)$$

$$2) R_b = L_d = 0 \Rightarrow$$

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(0 + 0) \cdot \min(|R_a|, |0 + 0|) \\ \approx s \cdot \text{sign}(R_a) \cdot 0 = 0 \quad (13)$$

$$3) |R_a| = |R_b| = +\mathbb{U}max \Rightarrow$$

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b) \cdot \\ \min(+\mathbb{U}max, +\mathbb{U}max) \\ \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b) \cdot +\mathbb{U}max \quad (14)$$

$$4) |R_a| = +\mathbb{U}max \wedge R_b = 0 \Rightarrow$$

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(0 + L_d) \\ \cdot \min(+\mathbb{U}max, |0 + L_d|) \\ \approx s \cdot \text{sign}(R_a) \cdot L_d \quad (15)$$

$$5) |R_a| = +\mathbb{U}max \Rightarrow$$

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b + L_d) \\ \cdot \min(+\mathbb{U}max, |R_b + L_d|) \\ \approx s \cdot \text{sign}(R_a) \cdot (R_b + L_d) \quad (16)$$

$$6) L_d = 0 \Rightarrow$$

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b + 0) \cdot \min(|R_a|, |R_b + 0|) \\ \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b) \cdot \min(|R_a|, |R_b|) \quad (17)$$

7)  $R_b = 0 \Rightarrow$

$$\begin{aligned} R_c &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(0 + L_d) \cdot \min(|R_a|, |0 + L_d|) \\ &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_d) \cdot \min(|R_a|, |L_d|) \end{aligned} \quad (18)$$

8) otherwise  $\Rightarrow$

$$R_c \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b + L_d) \cdot \min(|R_a|, |R_b + L_d|) \quad (19)$$

### B. Simplifications for the right bound lower node

With the definition in (4), the equation mapped to the lower right node  $d$  in Fig. 3 in simplified form is as follows:

$$R_d \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(|R_a|, |L_c|) + R_b \quad (20)$$

With  $Umax$  sufficiently large, it is save to assume that if  $R_b = \pm Umax$ , then  $R_b + f(R_a, L_c) \approx R_b$ .

By an exhaustive search of all possible special input combinations in (20), including the case when no node has a special value, we get the following list of simplified equations:

1)  $R_a = R_b = 0 \Rightarrow$

$$R_d \approx s \cdot \text{sign}(0) \cdot \text{sign}(L_c) \cdot \min(0, |L_c|) + 0 = 0 \quad (21)$$

2)  $R_b = L_c = 0 \Rightarrow$

$$R_d \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(0) \cdot \min(|R_a|, 0) + 0 = 0 \quad (22)$$

3)  $|R_a| = |R_b| = +Umax \Rightarrow$

$$\begin{aligned} R_d &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \\ &\quad \min(+Umax, |L_c|) + R_b \\ &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(R_b) \cdot L_c + R_b \\ &\approx R_b \end{aligned} \quad (23)$$

4)  $|R_a| = +Umax \wedge R_b = 0 \Rightarrow$

$$\begin{aligned} R_d &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \\ &\quad \cdot \min(+Umax, |L_c|) + 0 \\ &\approx s \cdot \text{sign}(R_a) \cdot L_c \end{aligned} \quad (24)$$

5)  $|R_a| = +Umax \Rightarrow$

$$\begin{aligned} R_d &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \\ &\quad \cdot \min(+Umax, |L_c|) + R_b \\ &\approx s \cdot \text{sign}(R_a) \cdot L_c + R_b \end{aligned} \quad (25)$$

6)  $R_b = 0 \Rightarrow$

$$\begin{aligned} R_d &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(|R_a|, |L_c|) + 0 \\ &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(|R_a|, |L_c|) \end{aligned} \quad (26)$$

7) otherwise  $\Rightarrow$

$$R_d \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(|R_a|, |L_c|) + R_b \quad (27)$$

### C. Simplifications for the left bound upper node

Similar with the definition in (5), the equation mapped to the upper left node  $a$  in Fig. 3 in simplified form is as follows:

$$L_a \approx s \cdot \text{sign}(L_c) \cdot \text{sign}(L_d + R_b) \cdot \min(|L_c|, |L_d + R_b|) \quad (28)$$

With  $Umax$  sufficiently large, it is save to assume that if  $R_b = \pm Umax$ , then  $L_d + R_b \approx R_b$ .

By an exhaustive search of all possible special input combinations in (28), including the case when no node has a special value, we get the following list of simplified equations:

1)  $|R_b| = +Umax \Rightarrow$

$$\begin{aligned} L_a &\approx s \cdot \text{sign}(L_c) \cdot \text{sign}(L_d + R_b) \cdot \min(|L_c|, |L_d + R_b|) \\ &\approx s \cdot \text{sign}(L_c) \cdot \text{sign}(R_b) \cdot \min(|L_c|, +Umax) \\ &\approx s \cdot \text{sign}(R_b) \cdot L_c \end{aligned} \quad (29)$$

2)  $R_b = 0 \Rightarrow$

$$\begin{aligned} L_a &\approx s \cdot \text{sign}(L_c) \cdot \text{sign}(L_d + 0) \cdot \min(|L_c|, |L_d + 0|) \\ &\approx s \cdot \text{sign}(L_c) \cdot \text{sign}(L_d) \cdot \min(|L_c|, |L_d|) \end{aligned} \quad (30)$$

3) otherwise  $\Rightarrow$

$$L_a \approx s \cdot \text{sign}(L_c) \cdot \text{sign}(L_d + R_b) \cdot \min(|L_c|, |L_d + R_b|) \quad (31)$$

### D. Simplifications for the left bound lower node

Finally, the definition in (6), the equation mapped to the lower left node  $b$  in Fig. 3 in simplified form is as follows:

$$L_b \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(|R_a|, |L_c|) + L_d \quad (32)$$

By an exhaustive search of all possible special input combinations in (32), including the case when no node has a special value, we get the following list of simplified equations:

1)  $R_a = 0 \Rightarrow$

$$\begin{aligned} L_b &\approx s \cdot \text{sign}(0) \cdot \text{sign}(L_c) \cdot \min(|0|, |L_c|) + L_d \\ &= 0 + L_d = L_d \end{aligned} \quad (33)$$

2)  $|R_a| = +Umax \Rightarrow$

$$\begin{aligned} L_b &\approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(+Umax, |L_c|) + L_d \\ &\approx s \cdot \text{sign}(R_a) \cdot L_c + L_d \end{aligned} \quad (34)$$

3) otherwise  $\Rightarrow$

$$L_b \approx s \cdot \text{sign}(R_a) \cdot \text{sign}(L_c) \cdot \min(|R_a|, |L_c|) + L_d \quad (35)$$

## VI. PERFORMANCE ANALYSIS AND COMPARISON

### A. Methodology

An implementation of the proposed algorithm is validated using a simulation and compared against a textbook BP [7] algorithm. For the proposed algorithm, the simplified equations, as given in Sec. V-A to Sec. V-D, are implemented in the order given in this paper. If the condition is true, the accompanying simplified equation is executed and no further actions are taken for this node.

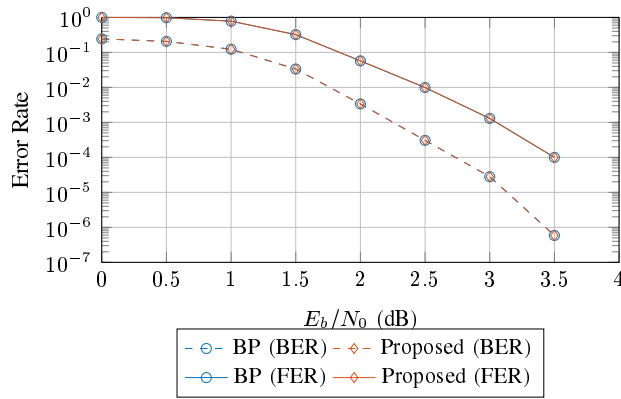


Fig. 4: BER and FER simulation results for the BP and proposed algorithms. A  $\mathcal{P}(1024, 512)$  polar code is.

To evaluate the performance of the proposed and other the decoder, a Monte Carlo simulation is performed. In the simulation, 10000 random vectors  $u$  are created, which are encoded into a  $\mathcal{P}(1024, 512)$  polar code codeword, using a systematic polar code encoder. Before transmission, the codeword is modulated with binary phase-shift keying (BPSK) and transmitted over an additive white Gaussian noise (AWGN) channel at noise levels (signal to noise ratio (SNR)).

Finally, the received signal is demodulated and the LLRs are computed. The LLRs are processed by the proposed and BP decoders, for which the maximum number of iterations for the proposed and BP algorithms are set to 30. After decoding the received signal, the estimated vector  $\hat{u}$  is compared with the send vector  $u$ , which finally results in frame error rate (FER) and bit error rate (BER) plots. At the same time several other statistics are gathered, like average iteration count, average latency and average operational counts like additions and comparisons.

*B. Proposed algorithm compared to the BP algorithm*

In both the simulated proposed and BP decoder the main scaling factor  $s = 0.9$  and the maximum number of iterations  $I_{max} = 30$  is used.

As shown in Fig. 4, both decoders have an equal bit error rate (BER) and frame error rate (FER) performance.

If we look at the decoding of a single codeword, it is trivial that, for any given SNR, the proposed algorithm with the simplified equations is performing less operations to decode a codeword, then the BP algorithm [7]. In Tab. I a breakdown of the number of operations is given, showing that the proposed algorithm performs approximately 23.6% less operation for a codeword send over a 0dB AWGN channel. A similar breakdown is given in Tab. II for a +4dB channel, resulting in approximately 27.2% less operations used in the proposed algorithm.

In Fig. 5 the average number of the specific operations are plotted for both the proposed and the BP decoder algorithms.

TABLE I: Breakdown of the average number of operations for the BP and the proposed algorithms at 0dB

	BP [7]	proposed	gain
# Iterations	$\approx 30$	$\approx 30$	0%
# Nodes	$\approx 583,668$	$\approx 583,668$	0%
Additions	$\approx 583,668$	$\approx 397,599$	$\approx -31.9\%$
Compares	$\approx 583,668$	$\approx 447,001$	$\approx -23.4\%$
Multiplications	$\approx 583,668$	$\approx 492,830$	$\approx -15.6\%$
Total # Ops	$\approx 1,751,004$	$\approx 1,337,430$	$\approx -23.6\%$

TABLE II: Breakdown of the average number of operations for the BP and the proposed algorithms at +4dB

	BP [7]	proposed	gain
# Iterations	$\approx 9$	$\approx 9$	0%
# Nodes	$\approx 180,104$	$\approx 180,104$	0%
Additions	$\approx 180,104$	$\approx 115,025$	$\approx -36.1\%$
Compares	$\approx 180,104$	$\approx 132,111$	$\approx -26.6\%$
Multiplications	$\approx 180,104$	$\approx 146,079$	$\approx -18.9\%$
Total # Ops	$\approx 540,312$	$\approx 393,215$	$\approx -27.2\%$

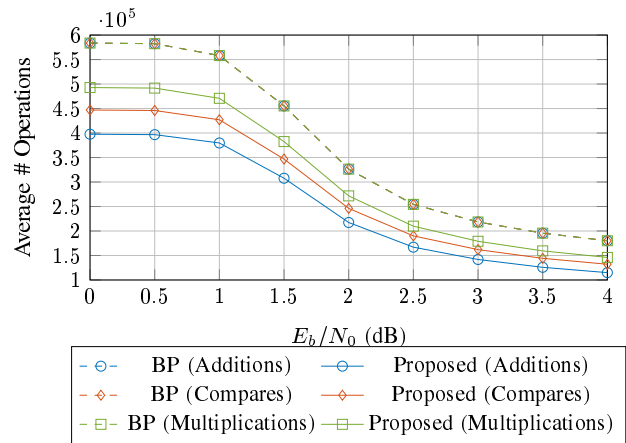


Fig. 5: BER and FER simulation results for the BP and proposed algorithms. A  $\mathcal{P}(1024, 512)$  polar code is.

TABLE III: Average frequency of occurrence of the proposed equations for  $R_c$

	Simplified equations from Sec.V-A			
	(12)	(13)	(14)	(15)
Count	25898 ( $\approx 18.7\%$ )	512 ( $\approx 0.4\%$ )	11610 ( $\approx 8.4\%$ )	1885 ( $\approx 1.4\%$ )
	Simplified equations from Sec.V-A			
	(16)	(17)	(18)	(19)
Count	1740 ( $\approx 1.3\%$ )	547 ( $\approx 0.4\%$ )	12963 ( $\approx 9.4\%$ )	83083 ( $\approx 60.1\%$ )

In Tab. III to Tab. VI, the average frequency of occurrence of the proposed equations are given.

After a more careful analysis of the used equations, a pattern became visible in which the proposed equations for the upper and lower nodes of the CU (see Fig. 1a) form a pair of equations. This holds for both computational directions in the iteration as well as for all simulated block lengths, i.e.  $N \in \{128, 512, 1024\}$ . After examination of all possible pairs, it is shown in Tab. VII and Tab. VIII that there is

TABLE IV: Average frequency of occurrence of the proposed equations for  $R_d$

		Simplified equations from Sec.V-B			
		(21)	(22)	(23)	(24)
Count		25898	512	11610	1885
		( $\approx 18.7\%$ )	( $\approx 0.4\%$ )	( $\approx 8.4\%$ )	( $\approx 1.4\%$ )
		Simplified equations from Sec.V-B			
		(25)	(26)	(27)	
Count		1740	12963	83630	
		( $\approx 1.3\%$ )	( $\approx 9.4\%$ )	( $\approx 60.5\%$ )	

TABLE V: Average frequency of occurrence of the proposed equations for  $L_a$

		Simplified equations from Sec.V-C		
		(29)	(30)	(31)
Count		11610	41769	100218
		( $\approx 7.6\%$ )	( $\approx 27.2\%$ )	( $\approx 65.2\%$ )

TABLE VI: Average frequency of occurrence of the proposed equations for  $L_b$

		Simplified equations from Sec.V-D		
		(33)	(34)	(35)
Count		26410	15360	111828
		( $\approx 17.2\%$ )	( $\approx 10.0\%$ )	( $\approx 72.8\%$ )

only a limited number of valid pairs. Therefore the number of required conditions is also limited, hence the possibility of a less complex implementation. Preliminary results, using the instruction set and cycle timings of an ARM Cortex-R4 processor, show that a net gain of  $\approx 13\%$  in the reduction of arithmetic operations used in the proposed algorithm compared to the usage of only the equation pairs (19)/(27) and (31)/(35), i.e. the textbook BP algorithm, could be expected.

### VII. CONCLUSION AND FUTURE WORK

In this paper we have introduced a number of simplifications for the equations used in the belief propagation (BP) algo-

TABLE VII: Valid equation pairs for right bound node computations.

Pair (upper/lower)	Condition
(12) / (21)	$R_a = R_b = 0$
(13) / (22)	$R_a = R_b = L_c = L_d = 0$
(14) / (23)	$ R_a  =  R_b  = +\mathbb{U}max$
(15) / (24)	$ R_a  = +\mathbb{U}max \wedge R_b = 0$
(16) / (25)	$ R_a  = +\mathbb{U}max$
(17) / (27)	$L_d = 0$
(18) / (26)	$R_b = 0$
(19) / (27)	otherwise

TABLE VIII: Valid equation pairs for left bound node computations.

Pair (upper/lower)	Condition
(29) / (34)	$ R_a  =  R_b  = +\mathbb{U}max$
(30) / (33)	$R_a = R_b = 0$
(30) / (34)	$ R_a  = +\mathbb{U}max \wedge R_b = 0$
(30) / (35)	$R_b = 0$
(31) / (34)	$ R_a  = +\mathbb{U}max$
(31) / (35)	otherwise

rihm. The proposed simplified algorithm results in an equal bit and frame error rate compared to BP [7]. In total, the proposed simplified algorithm, allows, in simulations, for the skipping of arithmetic operations upto 27.2%.

As future work we would like to explore the net benefits of the reduced number of operations on energy consumption in digital realizations and cycle counts on processors. Preliminary results show a cycle count reduction of 13% on an ARM cortex-R4 processor.

### ACKNOWLEDGMENT

This work is part of the research program Perspectief ZERO with project number P15-06 Project 4, which is (partly) financed by the Dutch Research Council (NWO).

### 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, 2009.
- [2] I. Tal and A. Vardy, "List Decoding of Polar Codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, may 2015.
- [3] 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.
- [4] G. Sarkis, P. Giard, A. Vardy, C. Thibault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, 2014.
- [5] E. Arıkan, "Systematic polar coding," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 860–862, 2011.
- [6] K. Niu, K. Chen, J. Lin, and Q. T. Zhang, "Polar Codes: Primary Concepts and Practical Decoding Algorithms," *IEEE Commun. Mag.*, vol. 52, no. July, pp. 192–203, 2014.
- [7] E. Arıkan, "Polar codes : A pipelined implementation," *4th Int. Symp. Broadband Commun. (ISBC 2010)*, pp. 11–14, 2010.
- [8] A. Elkelesh, M. Ebadat, S. Cammerer, and S. T. Brink, "Belief Propagation List Decoding of Polar Codes," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1536–1539, 2018.
- [9] C. Albayrak, C. Simsek, and K. Turk, "Low-complexity early termination method for rateless soft decoder," *IEEE Commun. Lett.*, vol. 21, no. 11, pp. 2356–2359, 2017.
- [10] Y. Ren, C. Zhang, X. Liu, and X. You, "Efficient early termination schemes for belief-propagation decoding of polar codes," *Proc. - 2015 IEEE 11th Int. Conf. ASIC, ASICON 2015*, 2016.
- [11] C. Simsek and K. Turk, "Simplified Early Stopping Criterion for Belief-Propagation Polar Code Decoders," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1515–1518, 2016.
- [12] S. Sun and Z. Zhang, "Architecture and optimization of high-throughput belief propagation decoding of polar codes," in *Proc. - IEEE Int. Symp. Circuits Syst.*, vol. 2016-July, IEEE, may 2016, pp. 165–168.
- [13] Y. S. Park, Y. Tao, S. Sun, and Z. Zhang, "A 4.68Gb/s belief propagation polar decoder with bit-splitting register file," *IEEE Symp. VLSI Circuits, Dig. Tech. Pap.*, pp. 2013–2014, 2014.
- [14] J. Sha, J. Liu, J. Lin, and Z. Wang, "A Stage-Combined Belief Propagation Decoder for Polar Codes," *J. Signal Process. Syst.*, vol. 90, no. 5, pp. 687–694, 2018.
- [15] A. B. Van Den Brink and M. J. Bekooij, "Conflict-Free Vectorized In-order In-place Radix-r Belief Propagation Polar Code Decoder Algorithm," in *ACM Int. Conf. Proceeding Ser.* New York, NY, USA: ACM, apr 2020, pp. 18–23. [Online]. Available: <https://dl.acm.org/doi/10.1145/3390525.3390539>
- [16] A. Eslami and H. Pishro-Nik, "On bit error rate performance of polar codes in finite regime," *2010 48th Annu. Allert. Conf. Commun. Control. Comput. Allert. 2010*, pp. 188–194, 2010.
- [17] Y. Ren, Y. Shen, Z. Zhang, X. You, and C. Zhang, "Efficient Belief Propagation Polar Decoder with Loop Simplification Based Factor Graphs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5657–5660, 2020.