APPROXIMATE BELIEF PROPAGATION DECODER FOR POLAR CODE

Menghui Xu, Shusen Jing, Chuan Zhang, Jun Lin, Weikang Qian and Xiaohu You

National Mobile Communications Research Laboratory, Southeast University, Nanjing, China
Email:{mhxu, shs.jing, chzhang, xhyu}@seu.edu.cn

ABSTRACT

Polar code is increasing its popularity recently for its capacity-achieving property for B-DMCs. However, when designing decoders for polar code, it has always been an inevitable concern for us to balance the decoding performance and the hardware consumption. In this paper, we propose an approximate belief propagation (BP) decoder for polar code for the first time. By introducing the approximate computing schemes, we reduced the critical path delay (CPD) and the hardware consumption of the conventional BP decoders. Simulation results show that the proposed approximate BP decoder achieves nearly the same decoding performance as the conventional one.

Index Terms— Polar code, belief propagation decoder, approximate computing, hardware consumption.

1. INTRODUCTION

Polar code, proposed by Arıkan's breakthrough paper [1],has received significant attentions from both academia and industries. It has been proved to be one of the first capacity-achieving codes for binary-input discrete memoryless channels (B-DMCs). Two main decoding algorithms for polar code are successive cancellation (SC) decoding (or SC list, SCL decoding) and belief propagation (BP) decoding. The SC algorithm decodes bits in serial schedule with low complexity, but it suffers from high decoding latency. On the other hand, BP decoder works much faster than SC decoder due to its inherent high parallelism and it is more popular for decoder implementation. However, faced with the performance requirements raised by applications that come along with next generation such as 5G wireless communication and Internet of Things (IoT), both polar decoders have to increase either their list size L (for SCL decoding) or iteration number I (for BP decoding). Since the complexities of SCL decoder and BP decoder are of $\mathcal{O}(LN\log N)$ [2–4] and $\mathcal{O}(IN\log N)$ [5], respectively, balancing the decoding performance and hardware implementation complexity has always been an inevitable concern for all designers.

However, 100% precision in computation is not always required, especially in some error-resilient systems such as image processing, handwritten recognition, and neuromorphic systems. In addition, decoders of forward error correction (FEC) codes can be viewed as an error-resilient system since they combat the noisy channel inputs. It has been shown in [6] that BP decoding algorithm can be implemented under noisy circuits. So decoders with approximate computing has been considered as a promising solution to achieve better decoding performance without hardware cost and energy dissipation. In prior works, approximate computing has been already implemented for LDPC decoders [7].

In this paper, we combined polar BP decoding and approximate computing together and proposed an efficient approximate BP decoder for polar code. The remainder of this paper is organized as follows. Section 2 briefly reviews polar codes and BP decoding algorithms. Section 3 presents the approximate BP decoder. Design flow for the proposed approximate BP decoder is given in Section 4. Section 5 compares the hardware consumption of architecture with the conventional one. Conclusions are drawn in Section 6.

2. REVIEW OF POLAR CODES AND BP DECODING ALGORITHM

2.A. Polar Codes

Any polar code can be determined by the parameter set of (N, K, A, u_{A^c}). Here, $N = 2^n$ denotes the code length, K is the number of information bits, A is the set of information bits, A^c is the complementary set of A, and u_{A^c} represents the frozen bits' values. For polar encoding, $x_1^N = (x_1, x_2, ..., x_{n-1}, x_n)$ is constructed based on source vector $u_1^N = (u_1, u_2, ...u_{n-1}, u_n)$ as follows:

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

where G_N and B_N are the generator matrix and bit reversal permutation matrix respectively, and $F^{\otimes n}$ is *Kronecker* power of $F = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$. For decoding, the estimate of u_1^N is obtained based on the channel output y_1^N. The reader is referred to [1] for more details.

2.B. BP Decoding Algorithm

Polar BP decoding can be illustrated by an n-stage factor graph [8] shown in Fig, (1). The factor graph is consisted of two types of nodes, F node and G node. Each node is labeled with coordinate (i, j). Here the parameter i indicates the stage number and the parameter j indicates the node number in the column. Polar BP decoder updates and passes messages iteratively through this factor graph.

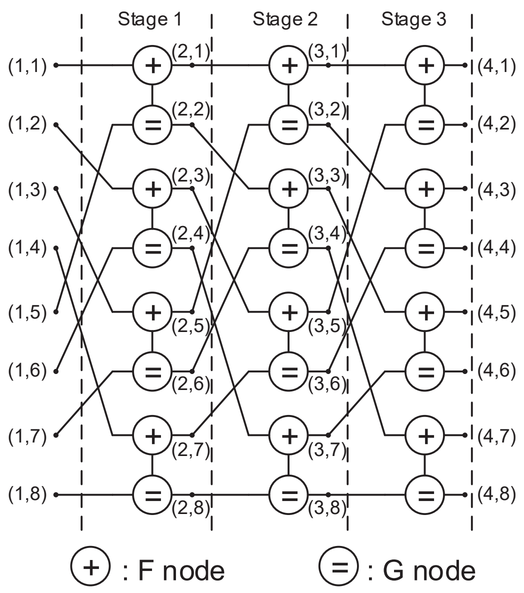

Fig. 1. Factor graph of BP decoding with $N = 8$.

For BP decoding, two types of log likelihood ratio (LLR) messages: left-to-right message L and right-to-left message R, are involved. They are initiated by Eqs. (2) and (3), respectively.

$$L_{n+1,j} = ln\frac{P(y_j|x_j = 0)}{P(y_j|x_j = 1)}, \tag{2}$$

$$R_{1,j} = \begin{cases} 0, & \text{if } j \in A, \\ \infty, & \text{if } j \in A^c. \end{cases} \tag{3}$$

Then messages are passed iteratively from left to right and then from right to left according to Eq. (4).

$$\begin{cases} L_{i,j} = f(L_{i+1,2j-1}, g(L_{i+1,2j}, R_{i,j+N/2})), \\ L_{i,j+N/2} = g(f(R_{i,j}, L_{i+1,2j-1}), L_{i+1,2j}), \\ R_{i+1,2j-1} = f(R_{i,j}, g(L_{i+1,2j}, R_{i,j+N/2})), \\ R_{i+1,2j} = g(f(R_{i,j}, L_{i+1,2j-1}), R_{i,j+N/2}); \end{cases} \tag{4}$$

where

$$f(x, y) = x + y, \tag{5}$$

$$g(x, y) \approx sign(x)sign(y)min(|x|, |y|). \tag{6}$$

After I iterations, the j-th bit is estimated according to:

$$\hat{u}_j = \begin{cases} 0 & \text{if } R_{1,j} \geq 0, \\ 1 & \text{else.} \end{cases} \tag{7}$$

3. PROPOSED APPROXIMATE BP DECODER

3.A. Quantization Schemes

Before giving details of BP polar decoders, we need to decide the quantization scheme first. We use $(64, 32)$ polar code for simulations. We find that the values of LLR mainly distributed in the interval of $[-35, 35]$ Which means 1 sign-bit and 5 integer-bits are required at least for quantization in order to achieve satisfying performance.

The fixed point simulation results for the $(64, 32)$ polar code with different quantization schemes are shown in Fig. (2). Here, $(s - k - l)$ denote s sign-bit, k bits integer and l bits fractional. According to Fig. (2), the (1-5-3) quantization scheme achieves a good trade off between performance and complexity, therefore is employed by following implementations.

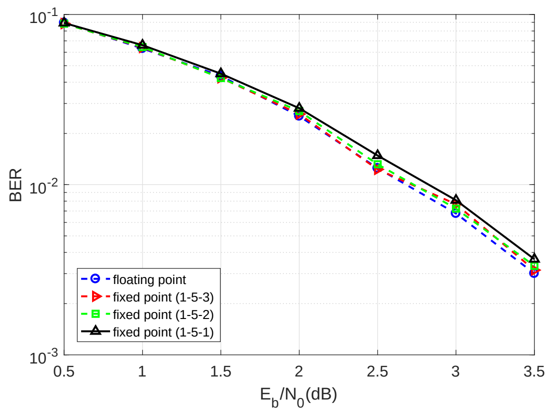

Fig. 2. Performance comparison of different quantization schemes.

3.B. Conventional Architecture for G Node

According to Eq. (6), it is shown that the comparison of absolute value is carried out during BP decoding process. As a result, the LLR messages are usually stored in sign-magnitude form (SMF) for convenience.

The major computation of the G node processing is magnitude comparison. For the conventional G node, we need to compare the two binary data from the high-order to the low-order bit by bit until two different bit with the same location are found. However, in some cases, we have to compare almost all the bits to find out which is the bigger or smaller one. But in fact, the two input data are about the same, which means whichever we choose to be the bigger or smaller one may not have an affection on the final result. So in these cases, the conventional architecture of G node results in a waste of time and power.

3.C. Proposed Approximate Architecture for G Node

In this section, an approximate architecture for G node is proposed. For approximate G node, we only compare the high-order $n-k$ bits and the rest k bits are ignored. The detail is shown in Fig. (3) with an example of $k = 2$.

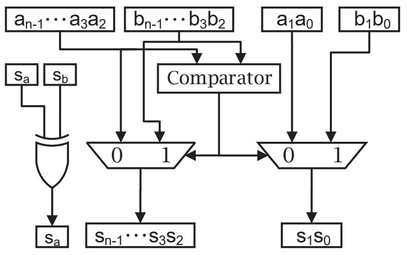

Fig. 3. Proposed approximate architecture for G node.

If $a_{[n-1:k]} \geq b_{[n-1:k]}$, then the approximate G node predicts $s_{[n-1:0]} = b_{[n-1:0]}$. Otherwise we have $s_{[n-1:0]} = a_{[n-1:0]}$. However, when we have $a_{[n-1:k]} = b_{[n-1:k]}$ and the ignored k bits of a is smaller that those of b, we will get a wrong result.

Supposing that a and b are random input numbers with uniform distribution, then the probability of $a_{[n-1:k]} = b_{[n-1:k]}$ and the probability of the $a_{[k-1:0]} < b_{[k-1:0]}$ are derived as follows:

$$\begin{cases} P(a_{[n-1:k]} = b_{[n-1:k]}) = (\frac{1}{2})^{n-k}, \\ P(a_{[k-1:0]} < b_{[k-1:0]}) = \frac{2^k-1}{2^{n+1}}. \end{cases}$$

Hence, the Error Rate (ER) of the proposed approximate G node is considered as follows:

$$ER = (\frac{1}{2})^{n-k} \cdot \frac{2^k-1}{2^{k+1}} = \frac{2^k-1}{2^{k+1}}. \tag{8}$$

According to Eq. (8), for a specific n, a larger k will cause greater performance loss and less hardware consumption. We use half-rate polar codes with code length $N = 64$ for simulation. The quantization scheme is (1-5-3), including 1 sign bit, 5 integer bits and 3 fractional bits. The result is shown in Fig. (4). With a proper number of ignored bits k, the approximate decoder performs nearly the same as the conventional accurate one with less hardware consumption.

3.D. Conventional Architecture for F Node

For the updating schedule in F nodes, messages should be transformed into their complement forms before addition or subtraction and then converted back to the SMF after the computation. As shown in Fig. (5), input messages a, b and output message s are stored in the SMF. Let S_a, S_b, S_s represent the sign of a, b and s, respectively. Let M_a, M_b, M_s represent the magnitude of a, b and s, respectively. The F

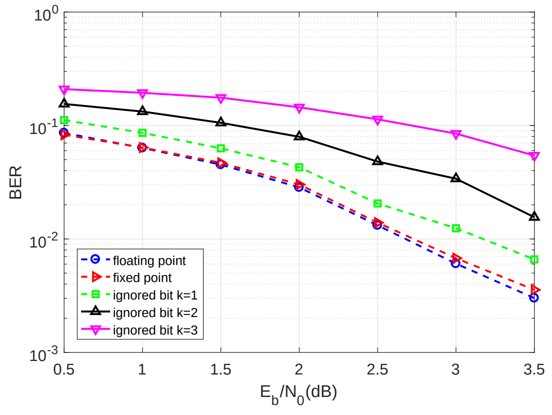

Fig. 4. Simulation results of BP decoders with different ignored bit k.

node functions as an adder and subtracter when $S_b = 0$ and $S_b = 1$ respectively. The adding one unit (AOU) adds 1 to its input. The architecture in Fig. (5) suffers from long critical path delay due to data format conversion at the input and output ports.

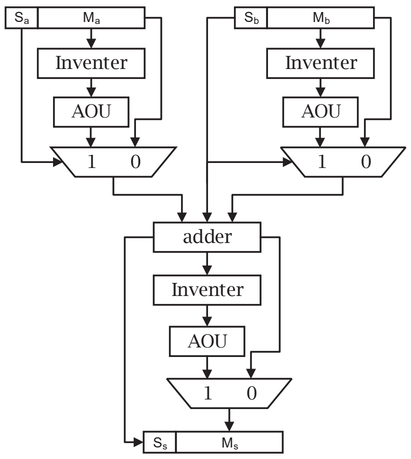

Fig. 5. Conventional architecture for F node.

3.E. Proposed Approximate Architecture for F Node

We notice that the architecture in Fig. (5) suffers from long CPD because of the data format conversion. In addition, the AOU and comparator in the conventional architecture increases the overall hardware consumption and the CPD. So, in this section, an efficient approximate architecture for F node is designed and is shown in Fig. (6).

As shown in Fig. (6), similarly, input messages a, b and output message s are also stored in the SMF. Let S_a, S_b, S_s represent the sign of a, b and s, respectively. Let M_a, M_b,

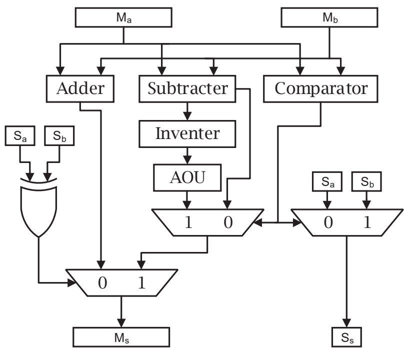

Fig. 6. Proposed approximate architecture for F node.

M_s represent the magnitude of a, b and s, respectively. For the proposed approximate architecture, subtraction is directly carried out instead of doing data format conversion before computation. As a result, we only need one data format conversion for F node. $M_a + M_b$ and $M_a - M_b$ are computed at the same time and the magnitude of the final result s is chosen from these two values.

Table 1. principles of the proposed approximate F node.

S_a	S_b	Relative Size	S_s	M_s
0	0	-	0	$M_a + M_b$
1	1	-	1	$M_a + M_b$
0	1	$M_a \geq M_b$	0	$M_a - M_b$
0	1	$M_a < M_b$	1	$-(M_a - M_b)$
1	0	$M_a \geq M_b$	1	$M_a - M_b$
1	0	$M_a < M_b$	0	$-(M_a - M_b)$

The specific arithmetical operations of F node is illustrated in table 1. When $S_a \oplus S_b = 0$, the F node implement $M_a + M_b$. Otherwise, it implements $M_a - M_b$. When $M_a \geq M_b$, $S_s = S_a$. Otherwise, $S_s = S_b$.

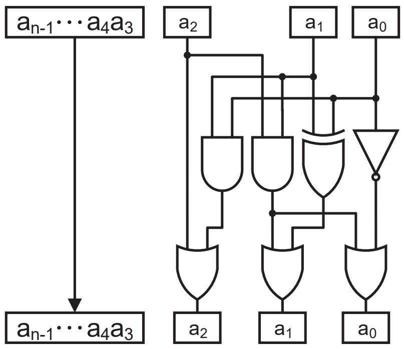

Fig. 7. Proposed approximate AOU with $m = 3$.

In the proposed architecture for F node, we also introduce approximate schemes for the add one unit (AOU) since the operation of AOU may not significantly affect the final computing result. We only add 1 to the m low-order bits of the input. In addition, the carry out bit of the adding one operation is dropped without being propagated to higher bits. To reduce the overall relative error, these m low-order bits are all set to 1 if the carry out bit is 1. Fig. (7) gives an example of the approximate AOU with $m = 3$.

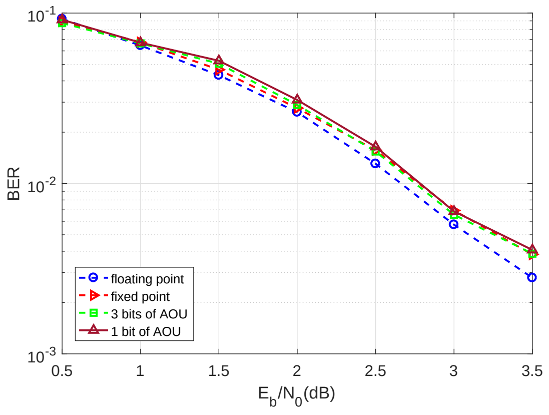

Fig. 8. Simulation results of BP decoders with different bits of AOU.

Similarly, we use half-rate polar codes with code length $N = 64$ for simulation. The quantization scheme is also(1-5-3). The result in Fig. (8) shows that even if when $m = 1$, the proposed approximate BP decoder still achieves almost the same decoding performance as the conventional one, which makes it possible to remove this AOU in the proposed architecture.

4. DESIGN FLOW OF APPROXIMATE POLAR BP DECODERS

Although by introducing approximate computing scheme, we reduced the hardware consumption of the conventional BP decoder, however, the decoding performance may suffer from significant degradation without careful design. In this section, we propose a design flow of the approximate polar BP decoder:

Step 1: For a polar BP decoder, we have to find optimal quantization schemes for left-to-right and right-to-left messages. For example, we use (5-3) for half rate polar codes with code length $N = 64$ bits.

Step 2: We introduce approximate computing scheme to the quantized polar BP decoder and redesign the F and G node respectively. By evaluating the simulation result of decoding performance, we make balance between the degree of accuracy and the hardware consumption.

Step 3: We combine all the approximate computing schemes together and adjust the degree of accuracy by the overall decoding performance.

5. HARDWARE IMPLEMENTATION AND COMPARISON

To demonstrate the advantage of the proposed approximate BP decoder, the corresponding implementation results for $(64-32)$ BP polar decoder are listed in table 2. It is shown that the proposed approximate BP decoder shows good hardware reduction than the accurate one. Compared with the conventional decoder, the arithmetic logic unit (ALUT) reduction of the approximate decoder is 36.3%. Meanwhile, the number of registers reduces 9.0%. However, this kind of hardware reduction has little adverse effect on decoding performance.

Table 2. implementation of different architectures for $(64-32)$ polar code.

Hardware overheads	Conventional	Approximate	Reduction
ALUT	$97,283$	$61,958$	36.3%
Registers	16214	14751	9.0%

6. CONCLUSION

In this paper, an approximate BP decoder for polar code is proposed for the first time. Approximate computing schemes are introduced to alleviate the contradiction between higher throughput and hardware consumption. The flow for designing an approximate polar BP decoder is also proposed. Simulation results show that the proposed approximate computing schemes lead to negligible performance degradation compared with the conventional one, which makes the proposed approximate polar BP decoder highly attractive in the future.

References

[1] Erdal Arıkan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” vol. 55, no. 7, pp. 3051–3073, 2009.

[2] Xiao Liang, Chuan Zhang, Menghui Xu, Shunqing Zhang, and Xiaohu You, “Efficient stochastic list successive cancellation decoder for polar codes,” in *Proc. IEEE International System-on-Chip Conference (SOCC)*, 2015, pp. 421–426.

[3] François Leduc-Primeau, Saied Hemati, Warren J Gross, and Shie Mannor, “A relaxed half-stochastic iterative decoder for LDPC codes,” in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, 2009, pp. 1–6.

[4] Chuan Zhang and Keshab Parhi, “Low-latency sequential and overlapped architectures for successive cancellation polar decoder,” vol. 61, no. 10, pp. 2429–2441, 2013.

[5] E. Arıkan, H. Kim, G. Markarian, U. Ozgur, and E. Poyraz, “Performance of short polar codes under ML decoding,” in *Proc. IEEE ICT-Mobile Summit*, Santander, Spain, Jun. 2009.

[6] Chu Hsiang Huang, Yao Li, and Lara Dolecek, “Belief propagation algorithms on noisy hardware,” *IEEE Transactions on Communications*, vol. 63, no. 1, pp. 11–24, 2015.

[7] Yangcan Zhou, Jun Lin, and Zhongfeng Wang, “Efficient approximate layered ldpc decoder,” in *IEEE International Symposium on Circuits and Systems*, 2017, pp. 1–4.

[8] Jin Sha, Xing Liu, Zhongfeng Wang, and Xiaoyang Zeng, “A memory efficient belief propagation decoder for polar codes,” *Communications, China*, vol. 12, no. 5, pp. 34–41, 2015.