Post-Processing Methods for Improving Coding Gain in Belief Propagation Decoding of Polar Codes

Shuanghong Sun, Sung-Gun Cho and Zhengya Zhang
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI, 48109-2122

***Abstract*—Belief propagation (BP) is a high-throughput, low-latency decoding algorithm for polar codes, but the error-correcting performance is known to be inferior than successive cancellation (SC) decoding. To improve the error-correcting performance of BP decoding, we design post-processing methods targeting false converged errors, oscillation errors, and unconverged errors that determine the performance of BP decoding. False convergence can be resolved by perturbing, or gradually freezing the information bits, followed by error cleanup using BP. Oscillations can be resolved by enhancing the stable bits and perturbing the unstable bits, followed by error cleanup using BP. Unconverged errors can be resolved by enhancing the reliably stable bits and weakening the unstable bits. Results show that the error rates of BP decoding can be improved by an order of magnitude or more, allowing it to overtake SC in error rate and coding gain. Post-processing can be implemented very efficiently, costing less than 4.3% overhead in silicon area, and it does not affect the throughput or latency of BP decoding.**

I. Introduction

Being the first provably capacity-achieving code for any binary-input discrete memoryless channels (B-DMC) [1], polar code holds great potential in its error-correcting capability. The two main decoding algorithms of a polar code are successive cancellation (SC) [1] and belief propagation (BP) [2]. SC decoding provides a lower error rate than BP decoding [3], but BP decoding offers a higher decoding throughput and a shorter latency. SC list [4] decoding offers the best error rate, but it suffers from an even lower throughput and a longer latency than SC decoding. For applications that require a multiple Gb/s data rate and latency well below 1 μs, BP decoding is possibly the best candidate available. However, the error-correcting performance of BP decoding of polar codes has been underwhelming and often does not even match BP decoding of LDPC codes of similar block lengths [5].

BP is an iterative message passing algorithm operating on a factor graph. BP gained its popularity in decoding LDPC codes. Despite its impressive performance in decoding LDPC codes, BP decoding of LDPC codes suffers from the error floor phenomenon [6], which sets the lower bound on achievable error rate. The majority of the errors in the error floor region is detectable by parity checks and can be corrected by post-processing methods [7]. Although polar codes do not use parity checks, decoding errors can be detected by a concatenated cyclic redundancy check (CRC) [8].

In this work, we formulate post-processing methods to improve the error-correcting capability of BP decoding. We demonstrate that the error rates of BP decoding can be lowered by at least an order of magnitude at a moderate to high SNR level, making BP decoding of polar codes more competitive than SC decoding. We show by chip synthesis that post-processing adds minimal overhead to BP decoding. A BP decoder with post-processor can be implemented in a substantially more cost-efficient manner, measured in Gb/s throughput per mm^2 silicon area.

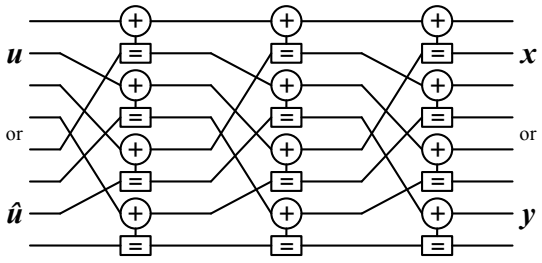

Fig. 1. Factor graph of N=8 polar code used by BP decoding.

II. Background

The generator matrix G_N of a polar code with block length $N = 2^n$ is the n-th Kronecker power of $F = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, i.e., $G_N = F^{\otimes n}$ [1]. Fig. 1 is the factor graph of an $N = 8$ polar code describing G_8, where $\oplus$ is XOR, and $\boxed{=}$ is pass-through. To obtain the codeword $\boldsymbol{x} = \boldsymbol{u}G$, the binary message $\boldsymbol{u}$ is passed from left to right of the factor graph, and the codeword is obtained on the right hand side. When the block length is sufficiently long, polarization [1] takes effect, referring to the phenomenon that bits become either highly reliable or highly unreliable under SC decoding. Highly reliable bits are used to carry information, and the highly unreliable ones are frozen, i.e., set to fixed values. The set of information bits is called the information set denoted by $\mathcal{A}$, and its complement $\mathcal{A}^c$ is the frozen set [1].

In SC decoding, the message bits $\hat{u}_0$ to $\hat{u}_{N-1}$ are decoded one after another based on the channel output $\boldsymbol{y}$ and the previously decoded bits, and the decoding latency is $\mathcal{O}(N)$. If $i \in \mathcal{A}^c$, $\hat{u}_i = 0$; otherwise $\hat{u}_i$ is decoded by the maximum likelihood decision rule:

$$\hat{u_i} = \begin{cases} 0 & \text{if } \frac{P(y,\hat{u}_0^{i-1}|u_i=0)}{P(y,\hat{u}_0^{i-1}|u_i=1)} > 1 \\ 1 & \text{otherwise} \end{cases}$$

In BP decoding, all the bits are decoded in parallel by passing soft LLR messages iteratively over the factor graph,

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

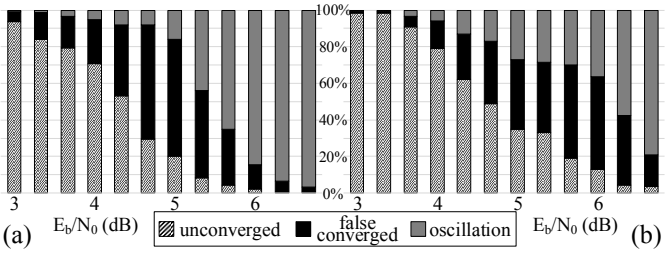

Fig. 2. BP error distribution of (a) a (256, 136) polar code and (b) a (1024, 522) polar code using a 6-bit quantized decoder.

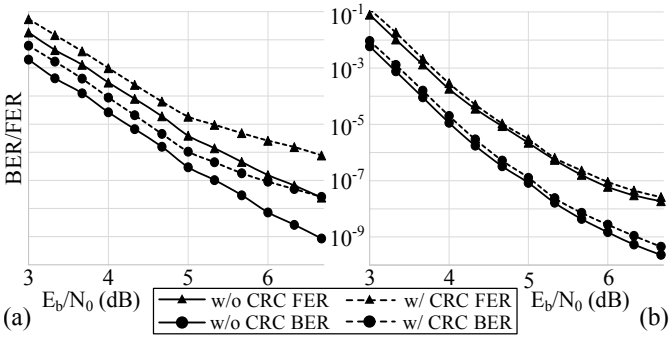

Fig. 3. Error rates of BP decoding with and without CRC concatenation: (a) a (256, 128) polar code and a (256, 136) polar code concatenated with 8-CRC, (b) a (1024, 512) polar code and a (1024, 522) polar code concatenated with 10-CRC.

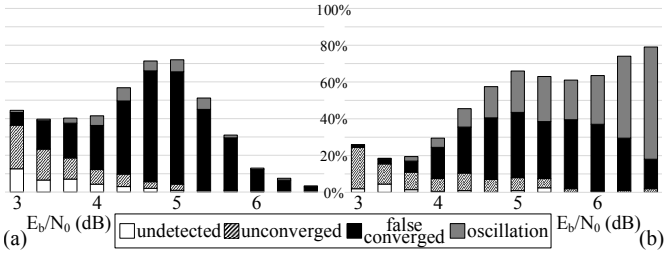

Fig. 4. BP error distribution with CRC concatenation and termination: (a) a (256, 128) CRC-concatenated polar code and (b) a (1024, 512) CRC-concatenated polar code that are decoded by a 6-bit quantized decoder.

from left to right and then right to left. The decoding latency is $\mathcal{O}(\log N)$. Information of the frozen set $\mathcal{A}^c$ is passed from left to right, and the channel LLRs $\boldsymbol{y}$ is propagated from right to left. Decision on the bits $\hat{\boldsymbol{u}}$ is obtained at the end of each iteration, and running more iterations generally improves the error-correcting performance.

III. Preliminaries

In our previous work [8], we classified BP decoding errors based on the patterns of hard decisions. Due to the lack of parity checks, it is not straightforward to detect errors. Therefore, we concatenate polar codes with CRC, making most of the BP decoding errors detectable. CRC also enables termination of BP decoding early [9] and in the right iteration. We briefly review the preliminaries in this section.

A. Types of Error

Fig. 2 shows the statistical breakdown of BP decoding errors for a (256, 136) and a (1024, 522) polar code. Although only two examples are shown, the breakdowns are representative. Error breakdown varies depending on the block length, the code rate, and the quantization [10], discussed in [8].

1) Unconverged Error: If hard decisions fail to agree within a set iteration limit, and there is no systematic pattern in the hard decisions, we classify it as an unconverged error. Unconverged errors dominate at low SNR.

2) False Converged Error: As SNR increases, it is more likely for BP decoding to reach convergence, however, BP decoding can converge to an incorrect codeword. We call it a false converged error.

3) Oscillation Error: At high SNR, BP decoding can result in oscillations, referring to the bit decisions bouncing back and forth following a systematic pattern over iterations. Oscillations are due to the loops in the factor graph that allow erroneous bits to propagate wrong messages that circle around in loops. The oscillation effect is often exacerbated by a finite numerical range that causes LLR messages to be clipped.

B. Concatenating with CRC

By monitoring hard decisions in consecutive iterations, unconverged errors and oscillation errors are detectable, but false converged errors are not. We reallocate parity bits by concatenating polar code with CRC to detect converged errors.

A CRC enables the detection of more than 99% of the errors in BP decoding. Reallocating a few frozen bits of polar code to parity bits of CRC increases the code rate of polar code, leading to a slight rise in error rate at the same SNR as shown in Fig. 3. The loss of coding gain is less noticeable for a longer code since the number of reallocated parity bits becomes negligible relative to the block length.

With CRC performing error detection, BP decoding can be terminated if CRC passes. Compared to the error breakdown in Fig. 2 using a fixed number of iterations, termination based on CRC is more reliable. CRC-based termination allows a decoder to lock in the correct codeword before the decoder diverges due to loopy message passing, helping to resolve some unconverged errors and oscillation errors as shown in Fig. 4. Although CRC-based termination reduces errors, the increment in code rate means a loss in effective SNR, which is the reason behind the rise in error rate. Also notice that CRC can miss errors, causing undetected errors, but the proportion of undetected error is negligible at moderate to high SNR.

IV. Post-Processing Methods

False converged errors and oscillation errors determine the error-correcting performance of BP decoding at moderate to high SNR level. A false converged error represents a steady state that "traps" the decoder. To escape, the steady state needs to be perturbed. Compared to a false converged error, an oscillation error provides direct clues as to which bits are stable and unstable. The insight can be exploited to stop oscillations by enhancing the stable bits and perturbing the unstable bits. Regular BP decoding can be used to clean up

the errors caused by perturbation. These intuitive ideas form the basis of our post-processing methods.

In BP decoding, the frozen set information is propagated from left to right of the factor graph. The frozen set information is carried by the so-called R messages. If a bit u_i is frozen, we set the R message of u_i to the maximum positive value, which effectively biases $\hat{u}_i$ to 0. If u_i is free, the R message of u_i is set to zero, so that $\hat{u}_i$ is unbiased and entirely determined by the extrinsic messages. In summary, if $i \in \mathcal{A}^c, R_i = M_{max}$; if $i \in \mathcal{A}, R_i = 0$. In the rest of the paper, we use the notation $M_{<sub>}$ to represent message magnitude, with the actual meaning defined where it is used.

In post-processing, we tune the R messages of the information bits to introduce perturbation. This could also be understood as biasing, or partially freezing the information bits towards one direction or another: a positive R message biases an information bit towards 0; and vice versa.

Since perturbation injects noise to the system, to quantify the effect of perturbation in post-processing, we define a cost function based on the decoded soft decisions.

$$C(\mathbf{x}) \triangleq \sqrt{\frac{\sum_{i\in\mathcal{A}}(M_{max} - \mathbf{x}_i)^2}{|\mathcal{A}|}},$$

where $\mathbf{x}$ is the vector of soft decisions of the decoded bits and $\mathbf{x}_i \in [-M_{max}, M_{max}]$, and $\mathcal{A}$ is the set of information bits. Assume an all-zero codeword and a quantized BP decoder, and M_{max} is the maximum magnitude of a soft decision. The cost function is essentially a measure of the normalized average distance between the decoded soft decisions $\mathbf{x}$ and the transmitted codeword.

A. Post-Processing False Converged Error

To fix false converged errors, we apply a small perturbation to destabilize the converged state. A balancing act is needed as perturbation increases the noise, and will likely cause errors to be made. Therefore the perturbation needs to be kept low, and perturbation should be applied discriminatingly. We use the soft decision of a bit as an indication of the reliability of the bit's hard decision: if the magnitude of the soft decision is high, the hard decision is most likely correct, and vice versa. Therefore, we enhance the reliable bits and perturb the unreliable bits at the same time. We use regular BP decoding to clean up the errors introduced by perturbation. The post-processing method is described in Algorithm 1.

The post-processing method starts by recognizing whether a bit is reliable. If the soft decision $\mathbf{x}_i$ reaches a set threshold $M_{threshold}$, indicating the decision being reliable, the bit is enhanced by setting R_i with a small magnitude M_0 in the same direction as the hard decision, i.e., the sign of $\mathbf{x}_i$. Setting R_i in the same direction as $\mathbf{x}_i$ will amplify its influence on neighboring bits in subsequent iterations. If a bit is unreliable because $\mathbf{x}_i$ is below $M_{threshold}$, the bit is perturbed by randomly setting R_i to either M_0 or $-M_0$ to bias the bit towards 0 or 1, respectively. (The notation $rand(\pm 1)$ in the algorithm refers to randomly picking 1 or -1.) The goal of the post-processing is to push the state out of false convergence.

Algorithm 1: Post-processing false converged errors

```
for iter_count = 1 to iter_limit do
    if CRC fails && hard decisions are consistent then
        for i ∈ A do
            if |x_i| > M_threshold then
                if R_i == 0 then
                    R_i = sign(x_i) × M_0
                else if |R_i| < M_limit then
                    R_i = sign(x_i) × |R_i| × c
                else
                    R_i = −sign(R_i) × M_0
            else
                if R_i == 0 then
                    R_i = rand(±1) × M_0
                else if |R_i| < M_limit then
                    R_i = −sign(x_i) × |R_i| × c
                else
                    R_i = −sign(R_i) × M_0
```

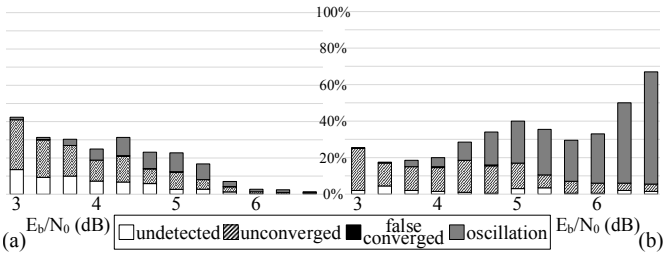

Fig. 5. Error distribution after post-processing false converged errors in BP decoding of (a) a (256, 128) CRC-concatenated polar code and (b) a (1024, 512) CRC-concatenated polar code.

After the perturbation, regular BP decoding is applied to clean up the errors introduced by the perturbation in an attempt to move towards convergence.

The post-processing is considered successful if either CRC is satisfied, indicating correct convergence (except for few numbers of undetected errors), or the hard decisions no longer remain consistent, indicating the decoding has escaped false convergence. If one attempt of post-processing is not successful, a second attempt using stronger enhancement and perturbation is applied ($c > 1$ in the algorithm). The attempts continue until a limit M_{limit} is reached to prevent excessive noise. In case M_{limit} is reached on a bit and the error is still trapped in false convergence, the bit is perturbed by applying a small bias M_0 in the opposite direction.

The error breakdown after applying post-processing to false converged errors is shown in Fig. 5. The post-processing resolves the majority of the false converged errors by BP decoding following perturbation. However, perturbation causes instability, turning some of the false converged errors into unconverged errors and oscillation errors.

The iteration-by-iteration plots of the cost function of BP

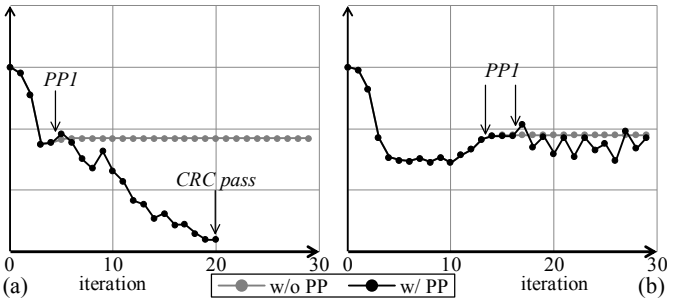

Fig. 6. Cost functions of BP decoding with and without applying post-processing Algorithm 1 (PP1): (a) an example of a false converged error resolved by Algorithm 1, and (b) an example of a false converged error that is not resolved by Algorithm 1.

decoding resulting in false converged errors are shown in Fig. 6. In the example illustrated in Fig. 6(a), the cost initially descends in BP decoding, and then false convergence is detected and post-processed at iteration 5. The cost first rises due to perturbation, allowing the decoder to escape false convergence. The following regular BP decoding cleans up the errors due to perturbation and converges to the correct codeword. In the example in Fig. 6(b), false convergence is detected and post-processed at iteration 14; and again detected and post-processed at iteration 17. Regular BP decoding follows each post-processing attempt, but the error turns into an unconverged error that cannot be solved by Algorithm 1.

B. Post-Processing Oscillation Error

In post-processing false converged errors, we used the magnitude of soft decisions to guide whether to apply enhancement or perturbation. Oscillation errors, on the other hand, provide direct clues of which bits are reliable and which ones are not. Unstable bits change their hard decisions periodically and are considered unreliable, and stable ones are consistent and considered reliable. To stop oscillations, stable bits are enhanced and unstable bits are perturbed. The post-processing method is described in Algorithm 2.

An oscillation error is detected by checking the consistency of hard decisions over consecutive iterations. Enhancement and perturbation are applied to the stable and unstable bits respectively using the similar approaches in Algorithm 1, starting by biasing the R messages by a small amount M_0 and letting regular BP decoding iterations clean up the errors. If post-processing is unsuccessful after one attempt, another attempt is used with stronger enhancement and perturbation until a biasing threshold of M_{limit} is reached.

The error breakdown after applying post-processing to both false converged errors and oscillation errors is shown in Fig. 7, where the vast majority of these two types of errors are resolved. Note that an error can evolve from one type to another in the post-processing procedure. If a false converged error or an oscillation error turns into an unconverged error, it cannot be solved by Algorithm 1 or 2.

The plots of the cost function of BP decoding resulting in oscillation errors are shown in Fig. 8. Fig. 8(a) illustrates an

Algorithm 2: Post-processing oscillation errors

for *iter_count* $= 1$ **to** *iter_limit* **do**
  if *oscillation is detected (of period* T*)* **then**
    for $i \in \mathcal{A}$ **do**
      if $sign(\mathbf{x}_i)$ *is consistent in* T *iterations* **then**
        if $R_i == 0$ **then**
          $R_i = sign(\mathbf{x}_i) \times M_0$
        else if $|R_i| < M_{limit}$ **then**
          $R_i = sign(\mathbf{x}_i) \times |R_i| \times c$
      else
        if $R_i == 0$ **then**
          $R_i = rand(\pm 1) \times M_0$
        else if $|R_i| < M_{limit}$ **then**
          $R_i = -R_i \times c$
        else
          $R_i = -sign(R_i) \times M_0$

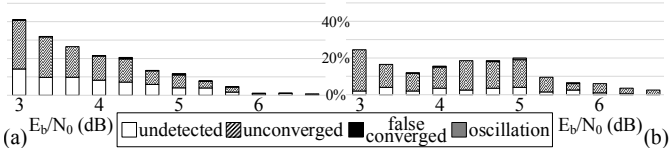

Fig. 7. Error distribution after post-processing oscillation errors and false converged errors in BP decoding of (a) a (256, 128) CRC-concatenated polar code and (b) a (1024, 512) CRC-concatenated polar code.

oscillation error. The error is detected and post-processed at iteration 9 using Algorithm 2. After a few iterations of regular BP decoding, the error is resolved. Fig. 8(b) illustrates a false converged error. False convergence is detected at iteration 7 and post-processed by Algorithm 1, and again at iteration 10. Following the second post-processing attempt, the error evolves to an oscillation error at iteration 16, and it is post-processed by Algorithm 2. Then it turns to another false converged error at iteration 19. Finally, the error is successfully resolved following a post-processing by Algorithm 1.

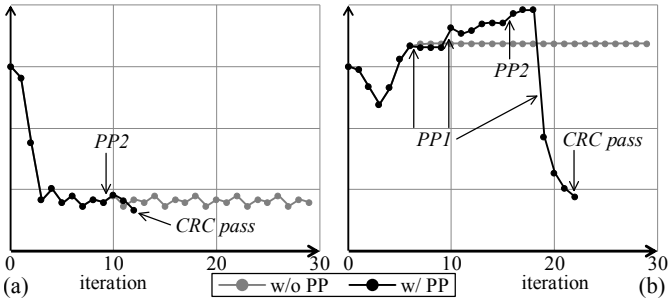

Fig. 8. Cost functions of BP decoding with and without applying post-processing Algorithm 1 and 2 (PP1 and PP2): (a) an example of an oscillation error resolved by Algorithm 2, and (b) an example of a false converged error evolving to an oscillation error that is resolved by Algorithm 1 and 2.

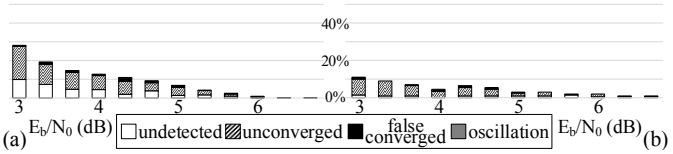

Fig. 9. Error distribution after post-processing unconverged errors, oscillation errors and false converged errors in BP decoding of (a) a (256, 128) CRC-concatenated polar code and (b) a (1024, 512) CRC-concatenated polar code.

C. Post-Processing Unconverged Error

With the vast majority of the false converged errors and oscillation errors removed, the only dominant errors left are unconverged errors. Many of the unconverged errors are in fact due to the perturbation applied to unreliable bits during post-processing. To resolve unconverged errors, the perturbation needs to be "unrolled" to enable convergence.

We can decide stable or unstable bits based on the consistency of the hard decision from one iteration to the next, and the magnitude of the soft decision. To resolve unconverged errors, we enhance the stable bits, but **weaken** the perturbation to the unstable bits to help convergence. The post-processing method is formulated in Algorithm 3.

Algorithm 3: Post-processing unconverged errors

```
for iter_count = 1 to iter_limit do
    if iter_count > iter_threshold && no oscillation is detected then
        for i ∈ A do
            if sign(x_i) is consistent over iterations && |x_i| > M_threshold then
                if R_i == 0 then
                    R_i = sign(x_i) × M_0
                else if |R_i| < M_limit then
                    R_i = sign(x_i) × |R_i| × c
            else if sign(x_i) is inconsistent over iterations then
                R_i = R_i / c
                if |R_i| < M_limit then
                    R_i = −R_i
```

An error is marked an unconverged error if BP decoding fails to converge after a sufficient number of BP iterations, and no oscillation is detected. The stable bits are enhanced using the same approach as in Algorithm 2. For the unstable bits, the perturbation is weakened by gradually reducing the bias in R messages, essentially undoing the perturbation. If a bit meets neither stable or unstable conditions, its R message will remain untouched. Most of the unconverged errors are resolved after applying Algorithm 3 as shown in Fig. 9.

The plots of the cost function of BP decoding resulting in unconverged errors are shown in Fig. 10. Fig. 10(a) illustrates an unconverged error, which is post-processed by Algorithm 3

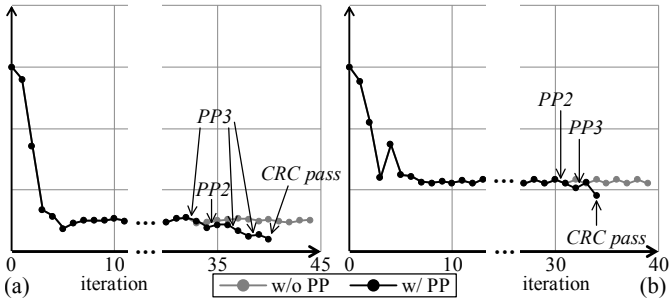

Fig. 10. Cost functions of BP decoding with and without applying post-processing Algorithm 2 and 3 (PP2 and PP3): (a) an example of an unconverged error resolved by Algorithm 2 and 3, and (b) an example of an oscillation error resolved by Algorithm 2 and 3.

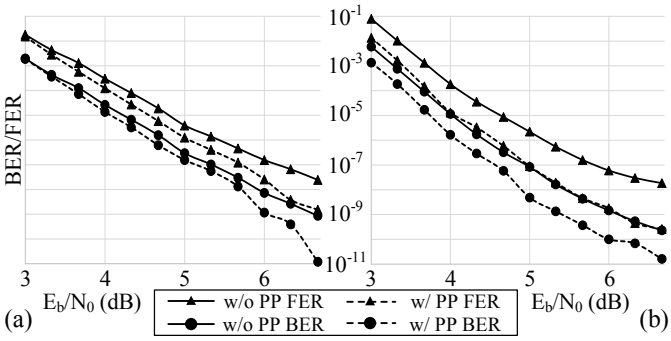

Fig. 11. Error rates of BP decoding with and without post-processing for (a) a (256, 128) polar code, and (b) a (1024, 512) polar code using 6-bit quantized decoders.

at iteration 33. The error evolves to an oscillation error at iteration 35, and is post-processed by Algorithm 2. Finally the error turns back to an unconverged error. After two post-processing attempts at iteration 37 and 39 by Algorithm 3, decoding converges. Fig. 10(b) illustrates an oscillation error, which is post-processed by Algorithm 2 at iteration 31. The error evolves to an unconverged error and resolved after post-processing using Algorithm 2 at iteration 33.

V. Results

The post-processing methods presented above can be efficiently implemented in BP decoding. Error detection is done by CRC and monitoring iteration-by-iteration hard decisions. If the decoding does not converge within an iteration limit, or if the decoding converges but fails CRC, an error is detected. Post-processing will only be applied to the detected errors.

The error-correcting results of post-processing are demonstrated in two code examples, a (256, 128) code and a (1024, 512) code, as shown in Fig. 11. Despite the decrease in coding gain due to CRC concatenation, post-processing easily recoup the loss, and improve the coding gain of the (256, 128) code by 0.8 dB at FER of 10^{-8} and the coding gain of the (1024, 512) code by more than 1 dB at FER of 10^{-8}.

With post-processing, BP decoding overtakes SC decoding in performance as shown in Fig. 12. The decoders are implemented in the same fixed-point quantization. The BP

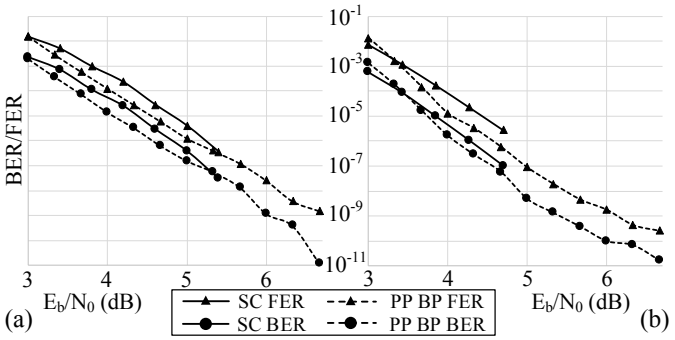

Fig. 12. Error rates of SC decoding and BP decoding with post-processing for (a) a (256, 128) polar code, and (b) a (1024, 512) polar code using 6-bit quantized decoders.

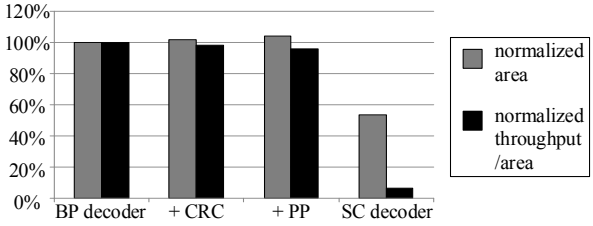

Fig. 13. Normalized silicon area and throughput/area ratio of different designs based on chip synthesis in a 45nm CMOS technology.

performance was obtained by FPGA emulation, while the SC performance was obtained by simulation. Due to the long latency in SC decoding and the slower software simulation, the SC error rate curves only extend to 10^{-6} in Fig. 12. In decoding the (256, 128) code and the (1024, 512) code, BP decoding with post-processing outperforms SC decoding by 0.4 dB and 0.5 dB, respectively, at moderate SNR.

The effectiveness and cost of implementing post-processing depends on the selection of parameters. Minimal parameter tuning is preferred as it ensures the compatibility of the three algorithms. We selected the smallest value for M_0 allowed by fixed-point quantization in the three algorithms to minimize the initial perturbation. All simulation results presented above were obtained using the same M_0, M_{limit} and c values, and they work effectively for the two codes across different SNR levels. Although we only presented the results of two polar codes, their decoding error types and error statistics are representative. Since the post-processing algorithms were designed for specific error types, rather than specific codes, the algorithms are expected to be suitable for any polar code.

To evaluate the implementing overhead, we performed chip synthesis in a 45nm CMOS technology, and the results are presented in Fig. 13. The baseline is a stage-parallel BP decoder [10]. A CRC codec adds a 1.8% area overhead, and post-processing costs an additional 2.5% area overhead. Since post-processing is conditionally invoked, it does not affect the average throughput and latency of BP decoding. A classic SC decoder uses half of the area as a stage-parallel BP decoder, but its throughput and latency are significantly worse. The figure of merit, in terms of throughput over area, of the BP decoder with post-processing is an order of magnitude better than an SC decoder as shown in Fig. 13. The better error-correcting performance, the higher throughput and lower latency are the key advantages of BP decoding with post-processing that make it more competitive than SC decoding.

VI. Conclusion

In this work, we present post-processing methods targeting false converged errors, oscillation errors, and unconverged errors to improve the error-correcting performance of BP decoding of polar codes. Post-processing is designed based on BP using modified frozen set information. For false converged errors, enhancement is applied to the bits of high reliability, and perturbation is applied to those of low reliability to escape false convergence. For oscillation errors, enhancement is applied to the stable bits to further strengthen these bits, and perturbation is applied to the unstable bits to stop oscillation. For unconverged errors, enhancement is applied to the stable bits, and perturbation to the unstable bits is unrolled to encourage convergence. In all three cases, BP decoding is used to clean up the errors introduced by perturbation.

Results show that post-processing of BP decoding improves the error rates by an order of magnitude or more at a moderate to high SNR level, demonstrating better error-correcting performance than SC decoding. Post-processing can be efficiently implemented in a BP decoder with negligible hardware overhead, and it does not affect the average throughput and latency, thereby making BP decoding more competitive than SC decoding for practical high-performance applications.

Acknowledgment

The authors would like to acknowledge the funding of this research by Intel and NSF CCF-1054270.

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, July 2009.

[2] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *Int. Symp. Wireless Commun. Syst.*, Nov 2011, pp. 437–441.

[3] B. Yuan and K. K. Parhi, "Architecture optimizations for BP polar decoders," in *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, May 2013, pp. 2654–2658.

[4] I. Tal and A. Vardy, "List decoding of polar codes," in *IEEE Int. Symp. Inf. Theory*, July 2011, pp. 1–5.

[5] N. Onizawa, T. Hanyu, and V. C. Gaudet, "Design of high-throughput fully parallel ldpc decoders based on wire partitioning," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 3, pp. 482–489, March 2010.

[6] T. Richardson, "Error floors of LDPC codes," in *Proc. Annu. Allerton Conf. Commun. Control and Computing*, vol. 41, no. 3, 2003, pp. 1426–1435.

[7] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. J. Wainwright, "Lowering LDPC error floors by postprocessing," in *2008 IEEE Global Telecommun. Conf.*, Nov 2008, pp. 165–168.

[8] S. Sun, S. G. Cho, and Z. Zhang, "Error patterns in belief propagation decoding of polar codes and their mitigation methods," in *2016 50th Asilomar Conf. Signals, Syst. and Comput.*, Nov 2016, pp. 1199–1203.

[9] Y. Ren, C. Zhang, X. Liu, and X. You, "Efficient early termination schemes for belief-propagation decoding of polar codes," in *2015 IEEE 11th Int. Conf. ASIC (ASICON)*, Nov 2015, pp. 1–4.

[10] S. Sun and Z. Zhang, "Architecture and optimization of high-throughput belief propagation decoding of polar codes," in *2016 IEEE Int. Symp. Circuits and Syst. (ISCAS)*, May 2016, pp. 165–168.