Lowering LDPC Error Floors by Postprocessing

Zhengya Zhang, Lara Dolecek*, Borivoje Nikolić, Venkat Anantharam, and Martin J. Wainwright
Department of EECS, University of California, Berkeley, CA 94720
*Department of EECS, Massachusetts Institute of Technology, Cambridge, MA 02139
zyzhang@eecs.berkeley.edu, dolecek@mit.edu, {bora, ananth, wainwrig}@eecs.berkeley.edu

***Abstract*–A class of combinatorial structures, called absorbing sets, strongly influences the performance of low-density parity-check (LDPC) decoders at low error rates. Past experiments have shown that a class of (8,8) absorbing sets determines the error floor performance of the (2048,1723) Reed-Solomon based LDPC code (RS-LDPC). A postprocessing approach is formulated to exploit the structure of the absorbing set by biasing the reliabilities of selected messages in a message-passing decoder. The approach converges quickly and can be efficiently implemented with minimal overhead. Hardware emulation of the decoder with postprocessing shows more than two orders of magnitude improvement in the very low bit error rate performance and error-floor-free operation below a BER of 10^{-12}.**

I. Introduction

Low-density parity-check (LDPC) codes have been demonstrated to perform very close to the Shannon limit when decoded iteratively [1]. Sometimes, excellent performance is only observed up until a moderate bit error rate (BER), because at lower BER the waterfall curve changes slope, leading to a so-called error floor [2]. Error floors are a major factor in limiting the deployment of LDPC codes in high-throughput applications, such as data storage and high-speed data communications.

In previous work [3], [4], we have developed an FPGA-based emulation system to investigate the causes of error floors. We showed that a class of combinatorial structures known as absorbing sets determines the error floor performance of LDPC codes and are due to the code construction. We also explored alternative quantization and decoder implementations that mitigate the effects of certain low-weight absorbing sets and lower the error floor [3], [5]. Nonetheless, the ultimate low error rate performance of an improved decoder is still dominated by absorbing sets.

This paper presents a general postprocessing approach that utilizes the graph-theoretic structure of absorbing sets. It carefully adjusts the appropriate messages in the iterative decoding once the decoder enters and remains in the absorbing set of interest. Compared to related work [6] – [8], we directly tackle the combinatorial structure of the absorbing set while minimizing the side effects. We are motivated by the work of Han and Ryan [9], but note that the proposed bi-mode syndrome-erasure decoding algorithm falls short of resolving the absorbing sets in some codes, where erasure decoding runs into stopping sets (which are defined in [10]) with high probability. Our proposed postprocessing approach is based on a message-passing algorithm with selectively biased messages. As a result, it can be seamlessly integrated with minimal overhead. Hardware emulation results show significant performance improvement at low error rates after postprocessing.

In Section II, we provide background on the decoding algorithm, a definition of absorbing sets, and results from Reed-Solomon based LDPC (RS-LDPC) decoder emulations. We then analyze a postprocessing approach capable of leaving the convergent state described by the absorbing set and converging to the right codeword. The results are presented in Section III. Section IV concludes the paper and considers directions for future work.

II. Background

A. Decoding Algorithm and Approximation

A low-density parity-check code is defined by a sparse $m \times n$ parity check matrix $\mathbf{H}$ where n represents the number of bits in the code block and m represents the number of parity checks. The $\mathbf{H}$ matrix of an LDPC code can be illustrated graphically using a factor graph, where each bit is represented by a variable node and each check is represented by a factor (check) node. An edge exists between the variable node i and the check node j if and only if $\mathbf{H}(j, i) = 1$.

Low-density parity-check codes are commonly iteratively decoded using the sum-product algorithm [1]. The algorithm operates on a factor graph, where soft messages are exchanged between variable nodes and check nodes. For suitably designed codes, convergence can usually be achieved within a small number of iterations. As an example, assume a binary phase-shift keying (BPSK) modulation and an additive white Gaussian noise (AWGN) channel. The binary values 0 and 1 are mapped into 1 and –1, respectively. In the first step of the algorithm, variable nodes x_i are initialized with the prior log-likelihood ratios (LLR) defined in (1) using the channel outputs y_i:

$$L^{pr}(x_i) = \log \frac{\Pr(x_i = 0 \mid y_i)}{\Pr(x_i = 1 \mid y_i)} = \frac{2}{\sigma^2} y_i, \qquad (1)$$

where σ^2 represents the channel noise variance.

Sum-product algorithm

Using a sum-product message-passing algorithm, the variable nodes send messages to the check nodes along the edges defined by the factor graph. The LLRs are recomputed based on the parity constraints at each check node and returned to the neighboring variable nodes. Each variable node then updates its decision based on the channel output and the extrinsic information received from all the neighboring check nodes.

978-1-4244-2324-8/08/$25.00 © 2008 IEEE.

The marginalized posterior information is used as the variable-to-check message in the next iteration. Variable-to-check and check-to-variable messages are computed using equations (2), (3), and (4).

$$L(q_{ij}) = \sum_{j' \in Col[i] \setminus j} L(r_{j'i}) + L^{pr}(x_i), \quad (2)$$

$$L(r_{ji}) = \Phi^{-1}\left(\sum_{i' \in Row[j] \setminus i} \Phi\left(\left|L(q_{i'j})\right|\right) \right) \left(\prod_{i' \in Row[j] \setminus i} \operatorname{sgn}\left(L(q_{i'j})\right) \right), \quad (3)$$

$$\Phi(x) = -\log\left(\tanh\left(\frac{1}{2}x\right) \right), \; x \geq 0. \quad (4)$$

The messages q_{ij} and r_{ji} refer to the variable-to-check and check-to-variable messages, respectively, that are passed between the i^{th} variable node and the j^{th} check node. In representing the connectivity of the factor graph, $Col[i]$ refers to the set of all the check nodes adjacent to the i^{th} variable node and $Row[j]$ refers to the set of all the variable nodes adjacent the j^{th} check node.

The posterior LLR is computed in each iteration using (5) and (6). A hard decision is made based on the posterior LLR as in (7).

$$L^{ext}(x_i) = \sum_{j' \in Col[i]} L(r_{j'i}), \quad (5)$$

$$L^{ps}(x_i) = L^{ext}(x_i) + L^{pr}(x_i), \text{ and} \quad (6)$$

$$\hat{x}_i = \begin{cases} 0, & if \quad L^{ps}(x_i) \geq 0 \\ 1, & if \quad L^{ps}(x_i) < 0. \end{cases} \quad (7)$$

The iterative decoding algorithm is allowed to run until the hard decisions satisfy all the parity check equations or when an upper limit on the iteration number is reached, whichever occurs earlier.

Approximate sum-product algorithm

Equation (3) can be simplified by observing that the magnitude of $L(r_{ji})$ is usually dominated by the minimum $|L(q_{i'j})|$ term. The update (3) can be approximated as

$$L(r_{ji}) = \min_{i' \in Row[j] \setminus i} \left|L(q_{i'j})\right| \prod_{i' \in Row[j] \setminus i} \operatorname{sgn}\left(L(q_{i'j})\right). \quad (8)$$

The magnitude of $L(r_{ji})$ computed using (8) is usually overestimated and correction terms are introduced to reduce the approximation error. The correction can be either in the form of a normalization factor or an offset shown as β in (9) [11].

$$L(r_{ji}) = \max\left\{ \min_{i' \in Row[j] \setminus i} \left|L(q_{i'j})\right| - \beta, 0 \right\} \prod_{i' \in Row[j] \setminus i} \operatorname{sgn}\left(L(q_{i'j})\right) \quad (9)$$

B. Absorbing Sets

Absorbing sets have been introduced in our previous work [3], [12], [13] to provide a characterization of certain types of decoding failure. An absorbing set is a particular kind of trapping set [2], with a precise combinatorial definition. Let $G = (V, F, E)$ be the bipartite graph associated with a parity check matrix $\mathbf{H}$, such that the set V corresponds to the columns of $\mathbf{H}$, the set F corresponds to the rows of $\mathbf{H}$, and $E = \{e(i, j) \mid \mathbf{H}(j, i) = 1\}$. Such a graph $G_{\mathbf{H}}$ is commonly referred to as the Tanner or factor graph of the parity check matrix $\mathbf{H}$ of a code. For a subset D of V, let $E(D)$ (resp. $O(D)$) be the set of neighboring vertices of D in F with even (resp. odd) degree with respect to D. With this setup we have the following:

Given an integer pair (a,b)*, an* (a,b) *absorbing set is a subset D of V of size a, with* $O(D)$ *of size b, and with the property that each element of D has strictly fewer neighbors in* $O(D)$ *than in* $F \setminus O(D)$*. We say that an* (a,b) *absorbing set D is an* (a,b) *fully absorbing set, if in addition, all variable nodes in* $V \setminus D$ *have strictly more neighbors in* $F \setminus O(D)$ *than in* $O(D)$*.*

A fully absorbing set, as defined above, can be understood as a special type of near-codeword [14] or trapping set [2], one which is stable under bit-flipping operations. An example of an (a,b) fully absorbing set with $a = 3$ and $b = 3$ is given in Fig. 1.

C. RS-LDPC Code and Emulation Results

Reed-Solomon based LDPC (RS-LDPC) codes [15] are regular, structured LDPC codes, with the girth being at least 6. The parity check matrix of this code family can be viewed as consisting of a two-dimensional array of permutation submatrices of equal size.

The focus of this paper is the (2048,1723) RS-LDPC code, which has column degree 6, row degree 32, and each component permutation submatrix is of size 64 × 64. This particular RS-LDPC has been adopted in the IEEE 802.3an 10GBASET standard [16]. The standard supports 10 Gb/s Ethernet over 100 meters of CAT-6a UTP (unshielded twisted-pair) cable. The high data rate is severely impaired by insertion loss, crosstalk, and interferences. The (2048, 1723) RS-LDPC code is selected specifically to provide sufficient coding gain to allow for a bit error rate (BER) performance of 10^{-12} or better [16].

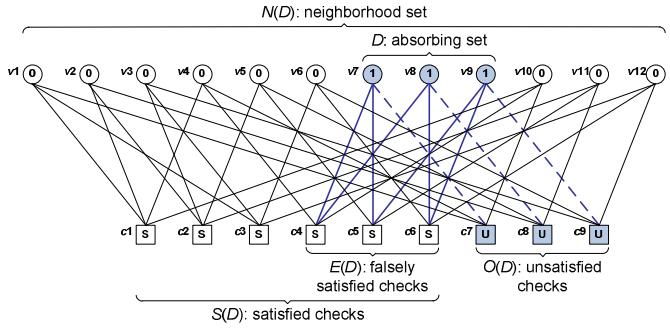


Figure 1. Illustration of a (3,3) fully absorbing set.

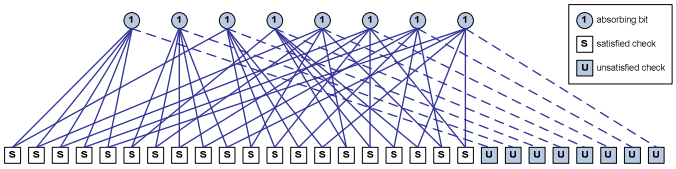


Figure 2. Illustration of a (8,8) absorbing set using the subgraph induced by the bits of the absorbing set.

We used high-throughput hardware emulation to explore the low error rate performance of the RS-LDPC code by analyzing the error traces. In well-designed decoders, we observed that a class of (8,8) fully absorbing set dominates the error floors. An illustration of the (8,8) absorbing set is shown in Fig. 2. Though only a subgraph induced by the variable nodes in the absorbing set is shown, all the (8,8) sets are indeed fully absorbing.

Even though this special (8,8) configuration is intrinsic to the code, and hence implementation-independent, its effect on BER is highly implementation-dependent. In particular, when the wordlength is finite, the effect of the absorbing sets can be exacerbated due to numerical saturation of messages. The approximate sum-product algorithm has a numerical advantage over the conventional sum-product algorithm because it eliminates message saturation caused by log-tanh estimations as in (9).

The performance comparison of different decoders is shown in Fig. 3 for a 6-bit Q4.2 (4 bits for integer and 2 bits for fraction) uniform-quantized sum-product decoder, along with a 4-bit Q4.0 approximate sum-product decoder and one with offset correction. The results are obtained using 20 decoding iterations. Despite a short 4-bit wordlength, the offset-corrected approximate sum-product decoder performs the best in the waterfall region. The error floor emerges at 10^{-10}, which still renders the implementation unacceptable for the 802.3an standard. In both the sum-product decoder and the approximate sum-product decoder, the (8,8) fully absorbing sets dominate the error floors.

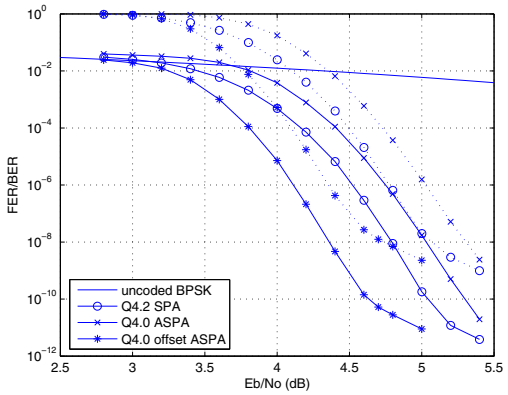


Figure 3. BER (solid line) and FER (dotted line) performance of a Q4.2 sum-product (SPA) decoder, a Q4.0 approximate sum-product (ASPA) decoder, and an ASPA decoder with offset correction for the (2048,1723) RS-LDPC code.

III. Postprocessing by Selective Biasing

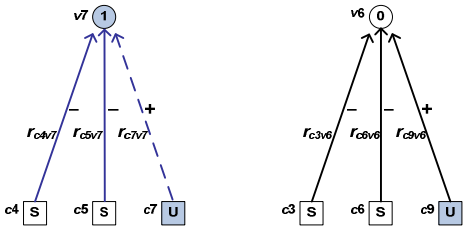
In this section, we describe a postprocessing method, based on the combinatorial structure of the (8,8) absorbing set, that dramatically lowers the error floor. The solution we provide is generally applicable to other absorbing sets, as demonstrated by resolving the (7,12) and the (11,6) absorbing sets. In the following discussion, we assume that the all-zeros codeword is used in transmission. Using the LLR definition in (1) and the hard decision rule in (7), the prior LLRs are nonnegative if received correctly, and the posterior LLRs are nonnegative if decoded correctly.

A. Selective Biasing

In an (8,8) absorbing set shown in Fig. 2, each bit in the absorbing set is connected to five satisfied checks (these checks are falsely satisfied because their neighboring bits are not all correct) and one unsatisfied check. The message from the unsatisfied check attempts to correct the wrong bit decision, as opposed to the messages from five falsely satisfied checks that reinforce the wrong bit decision. In a finite-wordlength decoder implementation, messages are easily saturated after several iterations. Consequently, soft decoding degenerates to a type of hard-decision decoding, in which the decisions are entirely based on majority counting, thus exacerbating the effects of absorbing sets.

An intuitive way in which to escape this absorbing state is to perform the following type of postprocessing: reduce the reliability of the messages from falsely satisfied checks, and increase the reliability of the message from the unsatisfied check. This selective biasing method alleviates the joint effect of a large number of incorrect messages.

As an illustration, consider a subgraph of Fig. 1, showing variable node $v7$ ($v7 \in D$) connected to falsely satisfied checks $c4$ and $c5$ ($c4, c5 \in E(D)$), as well as unsatisfied check $c7$ ($c7 \in O(D)$). We selectively bias messages r_{c4v7} and r_{c5v7} by reducing their reliabilities and bias the message r_{c7v7} by increasing its reliablity, so that r_{c7v7} reduces (or overcomes) the joint effect of r_{c4v7} and r_{c5v7}, thus reversing the wrong decision at $v7$. The messages are labeled in Fig. 4(a) by using "+" and "–" to indicate an increase in reliability (magnitude) and a decrease in reliability (magnitude), respectively.



(a) (b)

Figure 4. A subgraph of the (3,3) absorbing set showing the connectivity of (a) a bit in the absorbing set D and (b) a bit in the set $N(D)\backslash D$.

Despite the simplicity of this approach, its exact form is not implementable because the falsely satisfied checks cannot be separated from the correctly satisfied checks in a message-passing decoding process, and the post-processor needs to be further refined.

B. Relaxed Selectivity

Following the definition in Section II.B, let the neighborhood set $N(D)$ be the set of neighboring variable nodes to the unsatisfied checks in $O(D)$. Let $S(D)$ be the set of neighboring satisfied check nodes to the variable nodes in $N(D)$. Then $S(D)$ is composed of both the set of falsely satisfied checks $E(D)$ and the set of correctly satisfied checks. The example (3,3) absorbing set is annotated and shown in Fig. 1.

The set of unsatisfied checks $O(D)$ can be identified in each iteration of message-passing decoding, but no knowledge of the absorbing set D can be inferred besides that D is a subset of $N(D)$. Thus we relax the selectivity and increase the reliabilities of *all* messages from check nodes in $O(D)$ to variable nodes in $N(D)$ and decrease the reliabilities of *all* messages from check nodes in $S(D)$ to variable nodes in $N(D)$. As a result, in the (3,3) absorbing set illustrated in Fig. 1, each bit in D receives two weak messages from falsely satisfied checks and one strong message from the unsatisfied check, easing the absorbing state as expected. However, the relaxed selectivity causes each (correct) bit in $N(D)\backslash D$ to receive one strong message from the unsatisfied check and two weak messages from the satisfied checks as shown in Fig. 1 and the subgraph Fig. 4(b). The side effect is undesired, as it can cause the correct bits to reverse their values.

The biasing of the reliabilities needs to be carefully tuned, so that the incorrect bits within the absorbing set can be corrected, while the negative side effects are minimized. Two scenarios are depicted: Fig. 4(a) for a bit in an absorbing set ($v7 \in D$), and Fig. 4(b) for a bit in the neighborhood set but outside of the absorbing set ($v6 \in N(D)\backslash D$). The posterior LLRs of bits $v7$ and $v6$ are computed as in (10) and (11). The "+" and "–" signs indicate strengthened and weakened reliabilities. Assume uniform strengthening and weakening, referring to uniformly increase reliabilities to a fixed level, L_{strong}, in performing strengthening, and uniformly reduce reliabilities to a fixed level, L_{weak}, when performing weakening. Then the sum of extrinsic messages received at $v7$ and $v6$ are equal in magnitude but opposite in sign, as shown in (10) and (11).

$$\begin{aligned} L^{ps}(v7) &= L^{-}(r_{c4v7}) + L^{-}(r_{c5v7}) + L^{+}(r_{c7v7}) + L^{pr}(v7) \\ &= (L_{strong} - 2L_{weak}) + L^{pr}(v7) \end{aligned} \tag{10}$$

$$\begin{aligned} L^{ps}(v6) &= L^{-}(r_{c3v6}) + L^{-}(r_{c6v6}) + L^{+}(r_{c9v6}) + L^{pr}(v6) \\ &= -(L_{strong} - 2L_{weak}) + L^{pr}(v6) \end{aligned} \tag{11}$$

Let $L_{strong} - 2L_{weak} = \varepsilon$. Selecting a larger positive offset ε helps recovering the bits in the absorbing set, but also spreads more errors to the correct bits in the neighborhood set. An optimal selection of the ε value should be preferably small enough to control the spreading of errors to the correct bits while still capable of correcting the absorbing set errors. The selection criteria can be determined based on the prior LLRs.

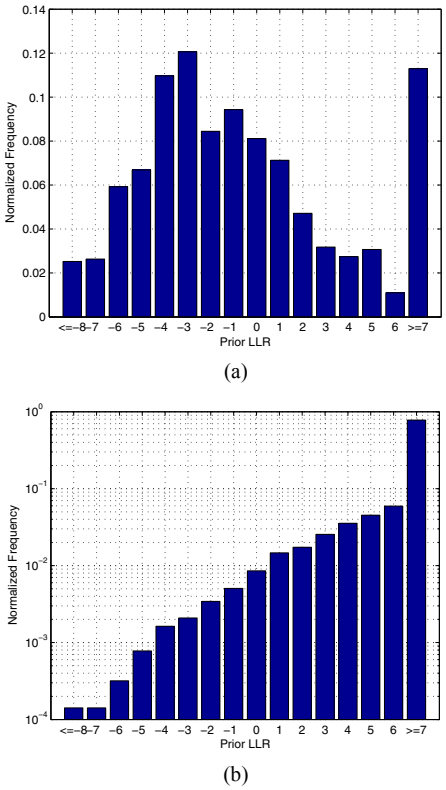
In the following, we reformulate the above analysis for the (8,8) absorbing set that dominates the error floor performance of the (2048,1723) RS-LDPC code. The cardinality of the neighborhood set $N(D)$ is 256. Each bit in $N(D)$ is connected to five satisfied checks and one unsatisfied check. We perform the following decoding steps:

1. Regular message-passing decoding
   Run for a fixed number of iterations. If decoding fails to converge, continue with the next step.

2. Postprocessing
   a. Message biasing: apply uniform strengthening to all messages from check nodes in $O(D)$ to variable nodes in $N(D)$ and uniform weakening to all messages from check nodes in $S(D)$ to variable nodes in $N(D)$.

   b. Follow up with a small number of iterations of regular message-passing decoding until postprocessing converges or declare failure.

The offset ε can be redefined as $\varepsilon = L_{strong} - 5L_{weak}$ with respect to the (8,8) absorbing set. By using hardware emulation 114 (8,8) absorbing set errors have been recorded together with the associated prior LLRs at an SNR level of 4.8 dB. The prior LLR distribution of bits belonging to the absorbing set D and bits belonging to the set $N(D)\backslash D$ are shown in Fig. 5(a) and (b).



(a)

(b)

Figure 5. Prior LLR distribution of (a) the bits in the absorbing set D, and (b) the bits in the set $N(D)\backslash D$. Results are based on 114 (8,8) absorbing set error traces of a (2048,1723) RS-LDPC decoder at an SNR level of 4.8 dB.

Based on Fig. 5(a), the prior LLRs of over half of the bits in the (8,8) absorbing sets are greater than –1; thus setting $\varepsilon = 1$, for instance, solves at least half of the errors and weakens the rest. Fig. 5(b) shows that the prior LLRs of the overwhelming majority of the bits in $N(D)\backslash D$ are very positive. Setting $\varepsilon = 1$ causes only 2.2% of the bits in $N(D)\backslash D$ (5 to 6 bits on average) to reverse their values and the remaining bits in $N(D)\backslash D$ are stable. Therefore, it is possible to select a small offset value ε to correct at least a reasonable fraction of the incorrect bits in the absorbing set, and affect negatively only a few correct bits.

After message biasing is applied, we follow up with a few more iterations of regular message-passing decoding to further break up the absorbing set and recover the few incorrectly flipped bits. An absorbing set usually collapses quickly after a fraction of bits in the absorbing set are corrected and the rest of the bits are weakened – the reinforcements between the bits of the absorbing set are significantly reduced and the absorbing set structure is resolved in a domino fashion.

C. Implementation and Results

We implemented the message biasing scheme through post-processing in a Q4.0 approximate sum-product decoder for the RS-LDPC code. The waterfall curve is free of error floor down to the BER level of 10^{-13}, which is as low as we can obtain using the BEE2 hardware emulation platform [17] with an average decoding throughput of 260 Mbps for ten days. The post-processor requires minimal overhead: the 4-bit wordlength is maintained and the approximate sum-product algorithm is unchanged. A small block of logic is added to perform weakening and strengthening operations on the corresponding messages after the decoder stalls.

The error count and weight statistics are shown in Table I. The average number of bitwise decoding errors per decoding failure converges to 8 at higher SNR levels, signifying the effect of (8,8) absorbing sets in determining the error floor performance. The post-processor is very effective in improving the error rate performance as seen in Fig. 6. The average error weight is larger after postprocessing, because the lower weight (8,8) absorbing sets are resolved and only the higher weight errors remain. In addition to (8,8) absorbing sets, the message biasing scheme successfully resolved (7,12) and (11,6) absorbing sets, which demonstrates the general applicability of the solution. In fact, we observe unresolved errors at 4.8 dB of SNR due to codewords of weights 14 and 16, which are undetected errors that even the optimal (albeit of high complexity) maximum likelihood decoding scheme would not be able to correct.

The average number of iterations for postprocessing is listed in Table II, showing approximately 3 iterations are required for the message biasing scheme to finally converge. The extra iterations for postprocessing can be easily accommodated due to faster convergence rates at a higher SNR level and infrequent invocation of the post-processor.

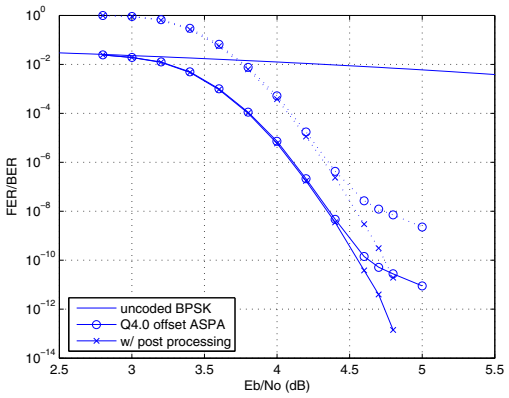


Figure 6. BER (solid line) and FER (dotted line) performance of a Q4.0 approximate sum-product decoder after postprocessing.

TABLE I
ERROR COUNTS AND WEIGHTS BEFORE AND AFTER POSTPROCESSING

SNR (dB)	Number of frame errors	Average weight	Number of frame errors after post-processing	Average weight
4.6	928	10.75	102	26.31
4.7	1271	8.60	32	26.38
4.8	733	8.02	2	15.00
5.0	196	8.01	0	-

TABLE II
AVERAGE NUMBER OF ITERATIONS FOR POSTPROCESSING

SNR (dB)	Average number of iterations for postprocessing
4.6	2.89
4.7	3.01
4.8	2.69
5.0	2.47

IV. CONCLUSIONS AND FUTURE WORK

Absorbing sets define a class of combinatorial structure that ultimately determines the error floor performance of some LDPC codes. The well-defined structure of absorbing sets allows us to design a post-processor to tackle them using the message-passing algorithm. The proposed message biasing scheme boosts the reliabilities of messages from unsatisfied checks and weakens the reliabilities of messages from the satisfied checks, so that the bits of the absorbing set can reverse the wrong values. The offset value ε is set small enough to minimize the negative impact on correct bits. The results show that an excellent low error rate decoding performance can be achieved with postprocessing due to the elimination of the absorbing set errors. The postprocessing algorithm converges quickly, usually within 3 or 4 iterations. The efficient implementation of this scheme allows us to easily pad it on an existing decoder with no change to the scheduling and only a minor fix to the decoding algorithm.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the contributions of the students, faculty, and sponsors of Berkeley Wireless Research Center and Wireless Foundations. This research was supported in part by NSF CCF grant no. 0635372, Marvell Semiconductor, Intel Corporation, Infineon Technologies, and the University of California MICRO program. NSF CNS RI grant no. 0403427 provided the computing infrastructure.

REFERENCES

[1] R.G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press, 1963.

[2] T. Richardson, "Error floors of LDPC codes," in *Proc. of the Allerton Conference on Communications, Control, and Computing*, Monticello, IL, pp. 1426-1435, Oct. 2003.

[3] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M.J. Wainwright, "Design of LDPC decoders for low error rate performance," submitted to *IEEE Trans. on Communications*, Mar. 2008.

[4] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M. Wainwright, "Investigation of error floors of structured low-density parity-check codes by hardware emulation," in *Proc. of IEEE GLOBECOM*, San Francisco, CA, Nov. 2006.

[5] Z. Zhang, L. Dolecek, M. Wainwright, V. Anantharam, B. Nikolic, "Quantization effects of low-density parity-check codes," in *Proc. IEEE Int. Conf. on Communications*, Glasgow, UK, Jun. 2007.

[6] S. Landner, O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. of IEEE Int. Conf. on Wireless Networks, Communications, and Mobile Computing*, Maui, HI, Jun. 2005.

[7] C.A. Cole, S.G. Wilson, "Message passing decoder behavior at low error rates," submitted to *IEEE Trans. on Communications*.

[8] A.I. Vila Casado, M. Griot, R.D. Wesel, "Informed dynamic scheduling for belief-propogation decoding of LDPC codes," in *Proc. IEEE Int. Conf. on Communications*, Glasgow, UK, Jun. 2007.

[9] Y. Han, W.E. Ryan, "LDPC decoder strategies for achieving low error floors," in *Proc. of Information Theory and Applications Workshop*, San Diego, CA, Jan. 2008.

[10] C. Di, D. Proietti, I.E. Telatar, T.J. Richardson, R.L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. on Information Theor*y, vol.48, no.6, pp. 1570-1579, Jun. 2002.

[11] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Communications*, vol. 53. no. 8, pp. 1288-1299, Aug. 2005.

[12] L. Dolecek, Z. Zhang, V. Anantharam, M.J. Wainwright, B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," submitted to *IEEE Trans. on Information Theory*, Feb. 2008.

[13] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, B. Nikolic, "Analysis of absorbing sets for array-based LDPC codes," in *Proc. of IEEE Int. Conf. on Communications*, Glasgow, UK, Jun. 2007.

[14] D. MacKay, M. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity check codes," *Electronic Notes in Theoretical Computer Science*, vol.74, pp. 97-104, 2003.

[15] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols," *IEEE Communications Letters*, vol. 7, no. 7, pp. 317-319, Jul. 2003.

[16] *Specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment 1: Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T*, IEEE Standard 802.3AN-2006, Sep. 2006.

[17] C. Chang, J. Wawrzynek, R.W. Brodersen, "BEE2: a high-end reconfigurable computing system," in *IEEE Design & Test of Computers*, vol.22, no.2, pp. 114-125, Mar. 2005.