Low Error Rate LDPC Decoders

Zhengya Zhang*, Lara Dolecek†, Pamela Lee‡, Venkat Anantharam‡, Martin J. Wainwright‡,
Brian Richards‡ and Borivoje Nikolić‡
*Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor
†Department of Electrical Engineering, University of California, Los Angeles
‡Department of Electrical Engineering and Computer Sciences, University of California, Berkeley

***Abstract*—Low-density parity-check (LDPC) codes have been demonstrated to perform very close to the Shannon limit when decoded iteratively. However challenges persist in building practical high-throughput decoders due to the existence of error floors at low error rate levels. We apply high-throughput hardware emulation to capture errors and error-inducing noise realizations, which allow for in-depth analysis. This method enables the design of LDPC decoders that operate without error floors down to very low bit error rate (BER) levels. Such emulation-aided studies facilitate complex systems designs.**

I. Introduction

Low-density parity-check (LDPC) codes can be designed to perform at rates extremely close to the Shannon limit [1], [2]. Since their recent rediscovery [3], LDPC codes have received wide-spread adoption in applications ranging from communications to data storage to improve both bandwidth efficiency and reliability. However, practical LDPC codes are not guaranteed to perform well at a low bit error rate (BER) [4] as required by some high-performance applications.

There is currently a shortage of simulation power and a lack of analytical approaches to fully understand the performance of practical LDPC codes at low error rates. As a result, high-throughput field-programmable gate array (FPGA) platforms have often been used for the purpose of verifying hardware architecture and accelerating code evaluation [4], [5].

This work demonstrates an LDPC decoder emulation platform that is constructed for studying decoder failure mechanisms at low error rate levels. The platform described in this work records noise realizations, as well as iteration-by-iteration states that lead to a decoding error. The noise realizations allow each error to be replicated in a Matlab reference model and interesting error profiles to be identified. More importantly, error statistics can be exploited in building an improved decoding algorithm.

The hardware emulation platform in described in Section II. The error traces captured by this platform lead to the classification of errors in Section III. The code-intrinsic absorbing errors are profiled in Section IV to clearly distinguish the effect of each implementation. Finally, error statistics are investigated in-depth for the formulation of an improved decoding algorithm to overcome the absorbing errors.

II. Hardware Emulation

An emulation-based design flow is developed for the investigation of LDPC codes, shown in Fig. 1. The design flow is based on the Berkeley Emulation Engine 2 (BEE2) platform

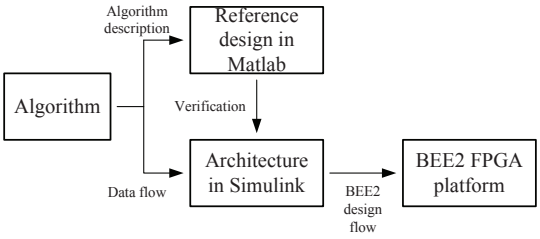


Fig. 1: Design flow for hardware emulation.

[6]. The BEE2 platform consists of an FPGA array hardware and a Simulink-based programming paradigm. The LDPC decoder is described in a fixed-point reference model in Matlab and its architecture is constructed in Simulink. The Simulink model is verified against the Matlab reference model before it is mapped to FPGA. More parallel decoder architectures can be implemented on an FPGA, enabling a throughput of at least several orders of magnitude higher than software simulations to reach very low BER levels. The Simulink-based design flow allows the rapid translation from data-flow-based design entry to hardware, facilitating an iterative design and refinement.

A. Design Flow

The decoder is hierarchically constructed in a bottom-up manner. The basic component modules are designed and verified in Simulink. The component modules are parameterized by wordlength and quantization. A Matlab script takes as input the **H** matrix of the LDPC code, instantiates modules from the design library and establishes connections between modules based on the **H** matrix. This approach simplifies the design process and enables design-time configurability.

B. Emulation Setup

Block RAMs on the FPGA record the noise realizations and final iterations of soft decisions when decoding fails. With a large number of block RAMs available on modern FPGA devices, a large memory bandwidth and fast access are possible. An on-chip PowerPC microprocessor controls the decoder by issuing start and stop commands, setting an upper limit on the number of decoding iterations, and adjusting the noise variance for different SNR levels. The hardware emulation platform is illustrated in Fig. 2. This platform allows the characterization of code performance and evaluation of practical implementation parameters [7]. Error traces captured

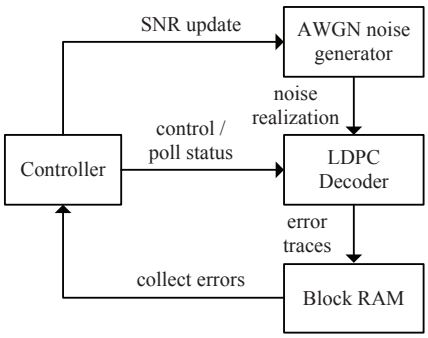


Fig. 2: An LDPC decoder emulation platform.

TABLE I: FPGA resource utilization

	Available	4-bit	6-bit	8-bit
Slice FF	66,176	9,153	10,017	10,881
4-input LUT	66,176	8,386	10,296	11,946
Slices	33,088	8,199	9,706	10,828
Total LUT	66,176	11,252	13,645	15,778
Block RAM	328	134	134	134

by the platform help in uncovering the causes of the error floors.

FPGA resource utilization is listed in Table I for the decoders (together with AWGN noise generators) of a (6,32)-regular (2048,1723) RS-LDPC code [8][1]. The decoders are implemented in a layered architecture [9] with wordlengths of 4, 6, and 8 bits using the sum-product message-passing decoding algorithm [1]. These decoders are made resource efficient – occupying approximately 1/3 of the available slices and block RAMs on a Xilinx Virtex-II Pro XC2VP70 FPGA. The Xilinx AWGN noise generators are incorporated at the cost of approximately 900 slices. The remaining resource on-chip can be used to store error traces. These decoders achieve a peak throughput of 480 Mb/s using a 100 MHz clock. Hardware emulation extends the BER measurement below 10^{-10} within hours. For comparison, an optimized implementation of the decoder in C provides a peak throughput of only 260 kb/s on an Intel Xeon 2.4 GHz microprocessor.

III. Error classification

Hardware emulation uncovers the flattening of the BER-SNR curve at low BER levels, known as error floors [4]. Most errors in the error floor region exhibit rather pronounced characteristics, either oscillatory or absorbing. Both types of errors start with a small number of bits that are received incorrectly. An oscillation error is unstable under the message-passing decoding: the number of incorrect bits increases with iterations of message passing and falls when enough incorrect bits are accumulated. The fluctuation of the number of incorrect bits occurs in a periodic fashion.

[1]Here and in the remainder of the paper (x,y)-regular (n,k) code refers to a code transforming k input bits into n coded bits, and is such that each coded bit participates in x parity checks, and each parity check is incident to y coded bits.

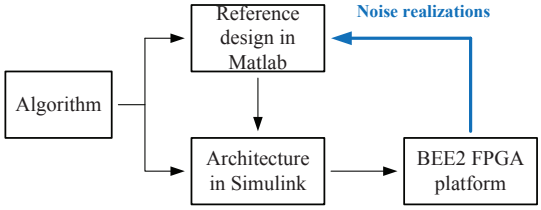


Fig. 3: Hardware emulation being used in a feedback loop.

Absorbing errors also start with a small number of bits that are received incorrectly. The values of these bits gradually stabilize, such that the incorrect bits remain incorrect and correct bits remain correct, thereby remaining in a local minimum state that absorbs the message-passing decoding algorithm.

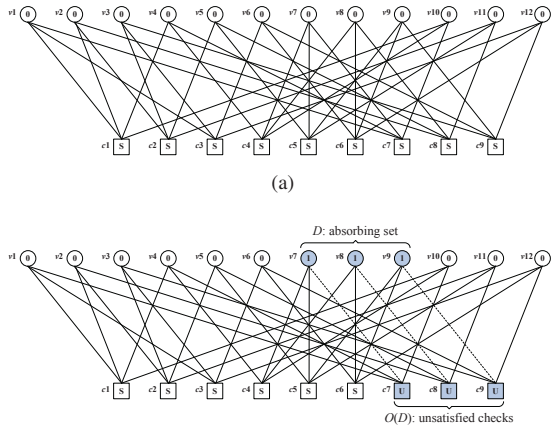
Hardware emulation accelerates code and decoder evaluation substantially. Very low error rate performance of practical LDPC decoders can be characterized and iteration-by-iteration soft decision capturing further allows the classification of errors that cause the decoder to fail. Hardware emulation can also be utilized as a step in an iterative design loop, as shown in Fig. 3. The input noise realizations that cause the decoding errors are captured and plugged in a functionally-equivalent decoder in Matlab to replicate each failure case. The error-replicating feedback simulation, when correlated with code structure and decoding algorithm, yields an in-depth understanding of the decoding errors.

A. Oscillation Errors and Absorbing Errors

Feedback simulation is applied in the investigation of decoding errors. Oscillation errors are unstable and can be interpreted as a consequence of the dynamics of quantized message exchange, while absorbing errors are the local minimum states and can be analyzed exactly.

Absorbing errors are related to the structure of the code under message-passing decoding. To illustrate an absorbing error, start with a factor graph such as the one in Fig. 4a that is associated with a small LDPC code (edges of the factor graph to the received output are not shown). Assume an all-zeros code is transmitted. By definition, all parity checks are satisfied. Suppose noise is injected to the transmitted bits, such that a subset of the bits are received incorrectly, e.g., the bits corresponding to variable nodes v_7, v_8, and v_9 now assume the incorrect value of 1 as shown in Fig. 4b. These incorrectly received bits cause some of the parity checks to be unsatisfied. With this setup, every incorrect bit receives two messages from the satisfied check nodes telling it to keep the incorrect decision, and it receives one message from the unsatisfied check node telling it to correct the decision. But two is more than one, and the incorrect bit cannot be recovered, thus the decoder is absorbed.

Absorbing sets were defined in the past [7], [10] to provide a valuable characterization of absorbing errors. In order to



(a)

(b)

Fig. 4: Illustration of (a) an all-zeros codeword transmitted, (b) a (3,3) fully absorbing set.

define an absorbing set, 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 $E = \{e(i,j)|\mathbf{H}(j,i) = 1\}$. Such a graph $G_{\mathbf{H}}$ is commonly referred to as the Tanner graph of the parity-check matrix $\mathbf{H}$ of a code, and is the relevant portion of the factor graph [11]–[13]. For a subset D of V, let $\mathcal{O}(D)$ be the set of neighboring vertices of D in F with 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 $\mathcal{O}(D)$ of size b, and with the property that each element of D has strictly fewer neighbors in $\mathcal{O}(D)$ than in $F \backslash \mathcal{O}(D)$. 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 fewer neighbors in $\mathcal{O}(D)$ than in $F \backslash \mathcal{O}(D)$. Using this notation, Fig. 4b shows a (3,3) absorbing set.

Two simple heuristic ways to characterize these sets are by weight (or a) and by stability. Everything else being equal, low-weight absorbing sets appear much more frequently when decoding fails. This phenomenon is more pronounced at higher SNR levels. The stability of an absorbing set is related to the structure of the set and the connectivity of the factor graph. In general, the ratio a/b provides clues to how stable an (a, b) absorbing set is – the higher the a/b ratio, the more stable the (a, b) absorbing set. Low-weight absorbing sets and more stable absorbing sets are of greater importance because they dominate the error floors.

IV. Error Profile

To demonstrate the applicability of the hardware emulation approach in identifying potentially important results, the finite-wordlength decoders of a (5,47)-regular (2209,1978) array-based LDPC code are studied in [7]. The class of array-based LDPC codes is known to perform well under iterative decoding [14]. This code is investigated by feedback simulation

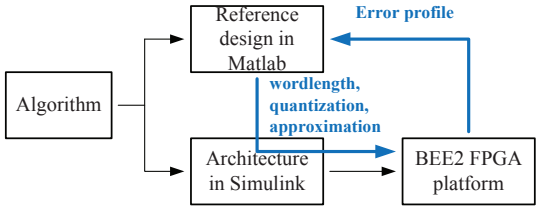


Fig. 5: Iterative improvement cycle by hardware emulation and feedback simulation.

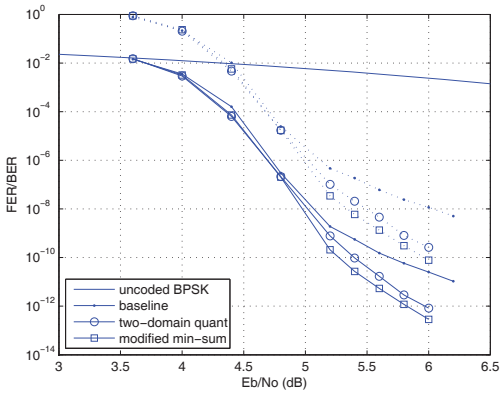


Fig. 6: FER (dotted lines) and BER (solid lines) performance of a (2209,1978) array-based LDPC code using 200 decoding iterations.

for the possible link between error events and the decoder implementation. The error profile of the baseline design is first characterized. The Matlab reference model of the decoder is then improved by tuning the decoder implementation. The improved design is subsequently mapped to FPGA for fast emulation and a complete new set of error profiles can be obtained for verification. The iterative investigation loop is illustrated in Fig. 5.

Emulation results are shown in Fig. 6. The baseline design is a 6-bit wordlength sum-product decoder implementation. The failures in the error floor region of this decoder are entirely due to absorbing errors. The error profile is shown in Fig. 7 demonstrating the dominance of (4,8) and (5,9) absorbing sets in causing the error floor due to their low weights.

The quantization of the baseline decoder is improved using a two-domain scheme with no increase in hardware complexity [7]. Fig. 6 shows that a two-domain quantized decoder performs better than the baseline in the error floor region. The absorbing error profile of the two-domain quantized decoder is compared to the baseline design in Fig. 7, which reveals the dominant absorbing sets (8,6) and (8,8) – higher weight and more stable than the (4,8) and (5,9) absorbing sets. These dominant absorbing sets account for the lower error floor and its steeper slope.

Min-sum approximation [15] is an alternative to the sum-product decoding algorithm. Min-sum approximation avoids

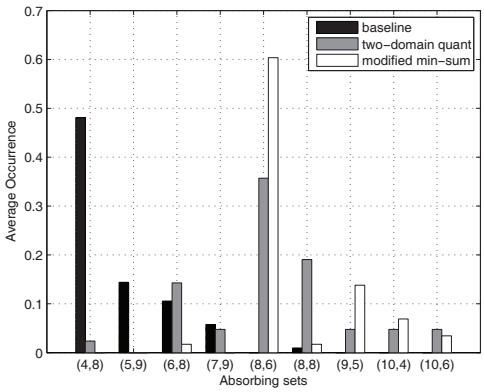


Fig. 7: Error profiles of three decoder implementations of a (2209,1978) array-based LDPC code at SNR=5.8dB.

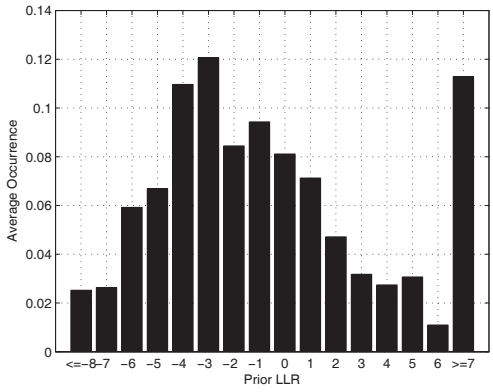


Fig. 8: Prior LLR distribution of the bits at absorbing set locations that eventually lead to an absorbing error.

some numerical saturation problems of a sum-product decoder implementation and it can perform better than the baseline in the error floor region. Its error profile shows that the dominant absorbing sets are (8,6) and (9,5) – higher weight and more stable than the (4,8) and (5,9) sets. Therefore the error floor is lower and its slope is steeper.

The results obtained on our hardware emulator are in excellent agreement with the stochastic based approach implemented in software, recently discussed in [16]. The approach in [16] utilizes a so-called importance sampling that samples from a distribution that is a tilted version of the original distribution, e.g., it is a mean-shifted Gaussian for the AWGN transmission. Under a properly chosen tilted distribution the decoding errors – typically very rare in the error floor of interest – appear significantly more frequently. This property drastically reduces the simulation time and enables a cross-check with the hardware results.

V. Error Mechanisms

Despite a lower error floor achieved with improved implementations, the underlying message-passing decoding algorithm is still a local algorithm when operating on graphs containing cycles – the error floor still remains and it is eventually defined by the stable absorbing sets. An algorithm improvement loop is formulated relying on the hardware emulation platform and feedback simulations. The statistics of the error-inducing channel outputs collected through hardware emulation are of the most interest as they shed light on certain "weaknesses" of the absorbing error mechanism. An improved algorithm can be designed to exploit the weaknesses.

The following discussions are based on a (2048,1723) RS-LDPC code [8]. Hardware emulation of this code uncovers the (8,8) absorbing sets as the dominant cause of the error floor [7]. Assume an all-zeros codeword is transmitted and $\{0,1\}$ are mapped to $\{1,-1\}$ in a BPSK modulation. Whenever an (8,8) absorbing error occurs, the emulation platform records the prior LLRs causing the error. Averaged over a large number of errors, the distribution of prior LLRs of the variable nodes at absorbing set locations can be obtained, as in Fig. 8 for statistics at the SNR level of 4.8dB. In this figure, the y-axis shows the average number of bits in an (8,8) absorbing set that assume each of the prior LLR values displayed on the x-axis. The center of the distribution falls slightly below 0, confirming that absorbing errors are mostly due to noise moderately out in the tail rather than noise values in the extreme tails.

This observation motivates a post-processing scheme that potentially lowers the error floor by orders of magnitude [17]. The scheme can be simply illustrated using the (3,3) absorbing set example in Fig. 4b. Each bit in the absorbing set is connected to two 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 two falsely satisfied checks that reinforce the wrong bit decision. A perturbation can be applied in post-processing to reduce the reliability of the messages from satisfied checks (note that the correctly satisfied and the falsely satisfied checks are indistinguishable), and increase the reliability of the message from the unsatisfied check. The perturbation helps the bits in the absorbing set to recover the correct values, but it also causes some undesirable disturbance to the bits outside of the absorbing set. Therefore the perturbation needs to be kept small and Fig. 8 confirms that a small perturbation is usually sufficient in helping most of the bits at absorbing set locations to recover the correct values and the remaining errors can be resolved by follow-up iterations of regular message-passing decoding.

The emulation results in Fig. 9 show that the error floor of the baseline design (a 4-bit wordlength, modified min-sum decoder) is removed after applying post-processing. The average bit error count per decoding failure in the baseline decoder converges to 8 at higher SNR levels, as shown in Fig. 10, signifying the importance of (8,8) absorbing sets in determining the error floor performance. The average error weight increases after post-processing, because the lower weight (8,8) absorbing sets are resolved and only higher weight errors

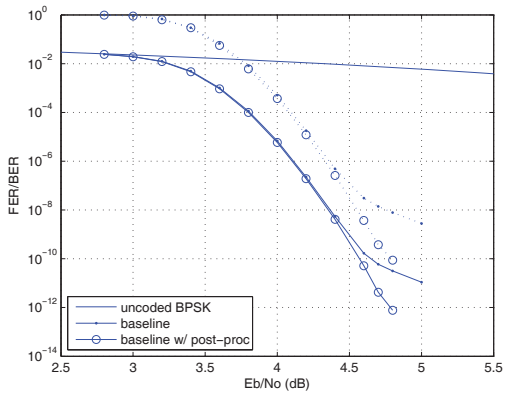


Fig. 9: FER (dotted lines) and BER (solid lines) performance of a (2048,1723) RS-LDPC code using 20 decoding iterations (and 10 iterations of post-processing where applicable).

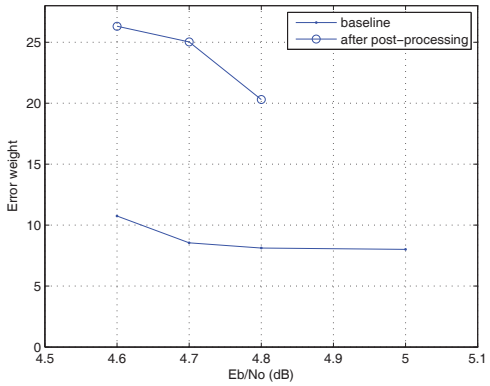


Fig. 10: Error weight before and after post-processing.

remain. A recent chip implementation incorporating post-processing demonstrates that most of the residual errors after post-processing are due to undetected errors [18], achieving near maximum-likelihood decoding performance.

VI. Conclusion

Hardware emulation plays an important role in the investigation and design of low error rate LDPC decoders. Error classification allows the separation of implementation-induced oscillation errors and code-intrinsic absorbing errors. Error profiling based on the absorbing set structure serves as the basis in characterizing and optimizing decoder implementations. The captured error statistics ultimately lead to the identification of weak links in the failure mechanism and a post-processing approach is formulated to lower the error floors.

Interesting future direction involves improving the code design from the point of view of the absorbing set spectrum. The data collected via the hardware emulator will be an indispensable component in designing a code with a better absorbing set distribution for an even lower error floor performance.

Acknowledgment

This research was supported in part by NSF CCF grant no. 0635372, Marvell Semiconductor and Intel Corporation through a UC MICRO grant. The design infrastructure was developed with the support of Center for Circuit & System Solutions (C2S2) Focus Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. The grant NSF CNS RI-0403427 provided the computing infrastructure.

References

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

[2] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[3] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

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

[5] L. Sun, H. Song, Z. Keirn, and B. Kumar, "Field programmable gate array (FPGA) for iterative code evaluation," *IEEE Transactions on Magnetics*, vol. 42, no. 2, pp. 226–231, Feb. 2006.

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

[7] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices," *IEEE Transactions on Communications*, vol. 57, no. 11, pp. 3258–3268, Nov. 2009.

[8] 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.

[9] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 684–698, Mar. 2006.

[10] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Transactions on Information Theory*, vol. 56, no. 1, Jan. 2010.

[11] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

[12] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, Linköping, Sweden, 1996.

[13] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, UK: Cambridge University Press, 2008.

[14] J. L. Fan, "Array codes as low-density parity-check codes," in *Proc. International Symposium on Turbo Codes and Related Topics*, Brest, France, Sep. 2000, pp. 543–546.

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

[16] L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolić, and M. J. Wainwright, "Predicting error floors of LDPC codes: deterministic bounds and estimates," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 908–917, Aug. 2009.

[17] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Communications Conference*, New Orleans, LA, Nov. 2008, pp. 1–6.

[18] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "A 47 Gb/s LDPC decoder with improved low error rate performance," in *Proc. Symposium on VLSI Circuits*, Kyoto, Japan, Jun. 2009, pp. 286–287.