Transactions Papers

Design of LDPC Decoders for Improved Low Error Rate Performance: Quantization and Algorithm Choices

Zhengya Zhang, *Student Member, IEEE,* Lara Dolecek, *Student Member, IEEE,*
Borivoje Nikolić, *Senior Member, IEEE,* Venkat Anantharam, *Fellow, IEEE,*
and Martin J. Wainwright, *Member, IEEE*

***Abstract*—Many classes of high-performance low-density parity-check (LDPC) codes are based on parity check matrices composed of permutation submatrices. We describe the design of a parallel-serial decoder architecture that can be used to map any LDPC code with such a structure to a hardware emulation platform. High-throughput emulation allows for the exploration of the low bit-error rate (BER) region and provides statistics of the error traces, which illuminate the causes of the error floors of the $(2048, 1723)$ Reed-Solomon based LDPC (RS-LDPC) code and the $(2209, 1978)$ array-based LDPC code. Two classes of error events are observed: oscillatory behavior and convergence to a class of non-codewords, termed absorbing sets. The influence of absorbing sets can be exacerbated by message quantization and decoder implementation. In particular, quantization and the log-tanh function approximation in sum-product decoders strongly affect which absorbing sets dominate in the error-floor region. We show that conventional sum-product decoder implementations of the $(2209, 1978)$ array-based LDPC code allow low-weight absorbing sets to have a strong effect, and, as a result, elevate the error floor. Dually-quantized sum-product decoders and approximate sum-product decoders alleviate the effects of low-weight absorbing sets, thereby lowering the error floor.**

***Index Terms*—Low-density parity-check (LDPC) code, message-passing decoding, iterative decoder implementation, error floor, absorbing set.**

Paper approved by T. M. Duman, the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received March 3, 2008; revised August 11, 2008 and November 20, 2008.

Z. Zhang was with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley and is now with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109 USA (e-mail: zhengya@eecs.umich.edu).

B. Nikolić, V. Anantharam, and M. J. Wainwright are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94720 USA (e-mail: {bora, ananth, wainwrig}@eecs.berkeley.edu).

L. Dolecek is with the Department of Electrical Engineering, University of California, Los Angeles (UCLA), Los Angeles, CA, 90095, USA (e-mail: dolecek@ee.ucla.edu).

This research was supported in part by NSF CCF grant no. 0635372, Marvell Semiconductor, Intel Corporation, and Infineon Technologies through the University of California MICRO program. NSF CNS RI grant no. 0403427 provided the computing infrastructure. This paper was presented in part at the IEEE Global Communications Conference (GLOBECOM), San Francisco, CA, November 2006, and in part at the IEEE International Conference on Communications (ICC), Glasgow, UK, June 2007.

Digital Object Identifier 10.1109/TCOMM.2009.11.080105

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); at a lower BER, the error curve often changes its slope, manifesting a so-called error floor [2]. Such error floors are a major factor in limiting the deployment of LDPC codes in high-throughput applications.

Exploring these error floors for realistic LDPC codes by software simulation on a general-purpose computer is not practical. Even an optimized decoder implemented in C and executed on a high-end microprocessor provides a peak throughput of only up to the order of 1 Mb/s. Consequently, months of simulation time would be required to collect at least tens of frame errors for a confident estimate of the BER at 10^{-10}. However, the use of field-programmable gate array (FPGA) platforms allows for substantial acceleration in the emulation of LDPC codes [2], [3].

This paper explores practical LDPC decoder design issues using an emulation-based approach. This investigation is motivated by Richardson's work on error floors [2], where he identified and semi-empirically defined a class of trapping sets using hardware emulation. Starting from the same point, we confirm some of these earlier findings, and moreover, we provide a combinatorial characterization of what we refer to as absorbing sets in terms of the graph structure of the code. For many LDPC codes, the associated factor graphs contain absorbing sets of lower weight than the minimum codeword weight. As a result, the performance of the code in the low error rate region is determined by the distribution and structure of the low-weight absorbing sets, rather than the minimum distance of the code [2], [4]. This paper sheds light on the effects of absorbing sets on the error floor levels in practical implementations of some LDPC decoders. Specifically, we advance the state-of-the-art in the following aspects: 1) the use of the absorbing set objects to quantify how the error counts are affected by wordlength, numerical quantization, and decoding algorithm choices; 2) differentiation of error mechanisms between oscillations and convergence to absorbing sets; 3) differentiation of weak from strong absorbing sets – weak absorbing sets can be eliminated by an optimal decoder implementation, while strong absorbing sets dominate the error floor of even an optimized decoder implementation; 4) propose dual quantization and demonstrate modified algorithms for improving the error floor performance by alleviating weak absorbing sets. We use high-performance hardware emulation throughout the investigation to uncover large datasets of error signatures and to verify conjectures. Compared to other related work [5]–[9], our study is based on the characterization of absorbing sets, which are classified by their structures. Most importantly, we analyze and provide intuition on why certain quantization choices and decoding algorithms perform better in the error floor region, thereby extending the definition of absorbing sets for practical usage.

In Section II, we provide background on the sum-product decoding algorithm, the quantization procedure, and decoder architecture of a family of high-performance regular LDPC codes. We present an implementation of the $(2048, 1723)$ Reed-Solomon based LDPC (RS-LDPC) [10] decoder which forms the basis of the hardware emulation platform. Error traces are collected from hardware emulations. In Section III, we analyze the error traces against the structure of the code to reveal the nature of error floors. In a decoder implementation with a sufficient wordlength, the hard decisions do not change after a number of decoding iterations while some parity checks remain unsatisfied. Such non-codeword errors are attributed to a class of combinatorial structures termed absorbing sets. We proceed with a series of experiments in Section IV using the $(2209, 1978)$ array-based LDPC code [11], which uncovers a collection of different absorbing sets in the error floor region. We develop methods to improve upon standard quantization approaches and experiment with alternative decoder implementations, thereby reducing the effects of weak absorbing sets and lowering the error floor.

II. LDPC Decoder Design and Emulation

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 usually 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 a concrete example, assume a binary phase-shift keying (BPSK) modulation and an additive white Gaussian noise (AWGN) channel. The binary channel bits $\{0, 1\}$ are represented using $\{1, -1\}$ for transmission over the channel. 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. This formulation assumes the information bits take on 0 and 1 with equal probability.

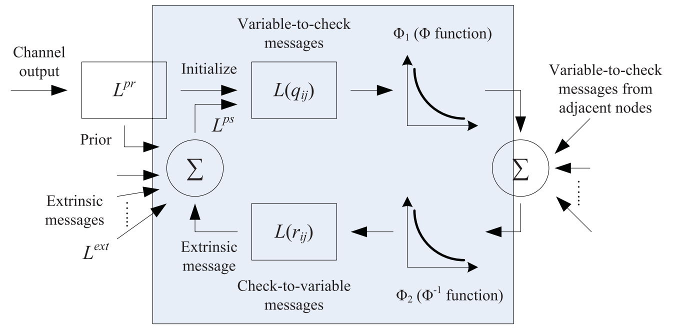

Channel output
L^{pr}
Initialize
Variable-to-check messages
$L(q_{ij})$
Φ_1 (Φ function)
Variable-to-check messages from adjacent nodes
L^{ps}
Prior
Σ
Extrinsic messages
L^{ext}
Extrinsic message
$L(r_{ij})$
Check-to-variable messages
Φ_2 (Φ^{-1} function)
Σ

Fig. 1. A sum-product message-passing decoder (one processing unit).

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

where σ^2 represents the channel noise variance.

1) Sum-product algorithm: Using a sum-product message-passing (belief propagation) 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. The marginalized posterior information is used as the variable-to-check message in the next iteration. A simplified illustration of the iterative decoding procedure is shown in Fig. 1. Variable-to-check and check-to-variable messages are computed using equations (2), (3), and (4).

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

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

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

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

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

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

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

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.

2) Approximate sum-product algorithm: Equation (3) can be simplified by observing that the magnitude of $L(r_{ij})$ is usually dominated by the minimum $|L(q_{i'j})|$ term. As shown in [12] and [13], the update (3) can be approximated as

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

Note that equation (8) precisely describes the check-node update of the min-sum algorithm. The magnitude of $L(r_{ij})$ 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 shown as α in (9) [5], an offset shown as β in (10) [5], or a conditional offset [6].

$$L(r_{ij}) = \frac{\min_{i' \in Row[j] \backslash i} |L(q_{i'j})|}{\alpha} \prod_{i' \in Row[j] \backslash i} \operatorname{sgn}(L(q_{i'j})). \quad (9)$$

$$L(r_{ij}) = \max\left\{ \min_{i' \in Row[j] \backslash i} |L(q_{i'j})| - \beta, 0 \right\} \times \prod_{i' \in Row[j] \backslash i} \operatorname{sgn}(L(q_{i'j})). \quad (10)$$

B. Message Quantization and Processing

Practical implementations only approximate the ideal realizations of the aforementioned algorithms. Such approximations are inevitable since real-valued messages can only be approximately represented, thus causing saturation and quantization effects, and moreover, the number of iterations is limited, so that the effectiveness of iterative decoding cannot be fully realized.

The approximations are illustrated by considering a pass through the sum-product decoding loop shown in Fig. 1. The channel output is saturated and quantized before it is saved as the prior LLR, L^{pr}. During the first phase of message passing, variable-to-check messages pass through the log-tanh transformation defined in (4), then the summation and marginalization, and finally the inverse log-tanh transformation. The log-tanh function is its own inverse, so the two transformations are identical. We refer to them as Φ_1 and Φ_2. The log-tanh function is approximated by discretization. The input and output of the function are saturated and quantized, thus the characteristics of this function cannot be fully captured, especially in the regions approaching infinity and zero.

In the second phase of message passing, the extrinsic messages L^{ext} are combined with the prior L^{pr} to produce the posterior probability L^{ps}. The prior, L^{pr}, is the saturated and quantized channel output; the extrinsic message, L^{ext}, is the sum of check-to-variable messages, which originate from the outputs of the approximated Φ_2 function. The messages incur numerical errors, and these errors accumulate, causing a decoder to perform worse than theoretically possible. The deficiencies due to real-valued implementations manifest themselves via performance degradation in the waterfall region, and a rise of the error floor.

The saturation and quantization effects are related to the fixed-point number format that is used in the processing and storage of data. We use the notation Q$m.f$ to represent a signed fixed-point number with m bits to the left of the radix point to represent integer values, and f bits to the right of the radix point to represent fractional values. Such a fixed-point representation translates to a quantization resolution of 2^{-f} and a range of $[-2^{m-1}, 2^{m-1} - 2^{-f}]$. Note that there is an asymmetry between the maximum and the minimum because 0 is represented with a positive sign in this number format. Values above the maximum or minimum are saturated, i.e., clipped. The wordlength of this fixed-point number is $m + f$. As an example, a Q4.2 fixed-point quantization translates to a quantization resolution of 0.25 and a range of $[-8, 7.75]$.

In an approximate sum-product implementation (8), Φ_1, summation, and Φ_2 are replaced by the minimum operation. The approximate algorithm introduces errors algorithmically, but it eliminates some numerical saturation and quantization effects by skipping through the log-tanh and the summation operations.

C. Structured LDPC Codes

A practical high-throughput LDPC decoder can be implemented in a fully parallel manner by directly mapping the factor graph onto an array of processing elements interconnected by wires. In this parallel implementation, all messages from variable nodes to check nodes and then in reverse are processed concurrently, yielding a complex, interconnect-dominated design. On the other hand, the memory bandwidth limits the throughput of a serial decoder [14]. A balance between throughput and memory bandwidth can be achieved if the underlying parity check matrix is regular and structured. The structure of the **H** matrix enables a parallel-serial architecture and a compact memory design.

Several known high-performance LDPC code constructions, including the Reed-Solomon based codes [10], array-based codes [11], as well as the ones proposed by Tanner *et al.* [15], share the same property that their parity check matrices can be written as a two-dimensional array of component matrices of equal size, each of which is a permutation matrix. Constructions using the ideas of Margulis and Ramanujan [16] have a similar property that the component matrices in the parity check matrix are either permutation or all-zeros matrices. In this family of LDPC codes, the $M \times N$ **H** matrix can be partitioned along the boundaries of $\delta \times \delta$ permutation submatrices. For $N = \delta\rho$ and $M = \delta\gamma$, column partition results in ρ column groups and row partition results in γ row groups. This structure of the parity check matrix proves amenable for efficient decoder architectures and recent published standards have adopted LDPC codes defined by such **H** matrices [17], [18].

D. Parallel-Serial Decoder Architecture for a Structured LDPC Code

In order to illustrate the decoder design, we select a $(6, 32)$-regular $(2048, 1723)$ RS-LDPC code. This particular LDPC code has been adopted as the forward error correction in the IEEE 802.3an 10GBase-T standard [18], which governs the operation of 10 Gb/s Ethernet over up to 100 m of CAT-6a unshielded twisted-pair (UTP) cable. The **H** matrix of this code contains $M = 384$ rows and $N = 2048$ columns. This matrix can be partitioned into $\gamma = 6$ row groups and $\rho = 32$ column groups of $\delta \times \delta = 64 \times 64$ permutation submatrices. We design a resource-efficient and configurable architecture to map the decoder to the FPGA emulation platform. Resource efficiency in the decoder design allows for more block RAMs to be allocated on-chip to capture soft traces for analysis; and with a configurable architecture, the decoder can be easily adapted to different codes in a short design cycle. The resulting parallel-serial architecture resembles a partially-parallel architecture [19], but we limit the parallelism by partitioning the **H** matrix in only one direction (i.e., parallelize among column partitions and process rows serially) to reduce complexity. Each of the partitions is configurable based on the structure of the **H** matrix. Compared to a fully parallel architecture [20], which is not configurable, or a fully serial architecture, which lacks the throughput [14], this parallel-serial design represents a tradeoff for the purpose of code emulation.

We apply column partition to divide the decoder into 32 parallel units, where each unit processes a group of 64 bits. Fig. 2 illustrates the architecture of the RS-LDPC sum-product decoder. Two sets of memories, $M0$ and $M1$, are designed to be accessed alternately. $M0$ stores variable-to-check messages and $M1$ stores check-to-variable messages. Each set of memories is divided into 32 banks. Each bank is assigned to a processing unit that can access them independently. In a check-to-variable operation defined in (3), the 32 variable-to-check messages pass through the log-tanh transformation, and then the check node computes the sum of these messages. The sum is marginalized locally in the processing unit and stored in $M1$. The stored messages pass through the inverse log-tanh transformation to generate check-to-variable messages. In the variable-to-check operation defined in (2), the variable node inside every processing unit accumulates check-to-variable messages serially. The sum is marginalized locally and stored in $M0$. This architecture minimizes the number of global interconnects by performing marginalization within the local processing unit.

The parallel-serial architecture allows efficient mapping of a practical decoder. For example, an RS-LDPC code of up to 8kb in block length can be supported on a Xilinx Virtex-II Pro XC2VP70 FPGA [21]. This architecture is also reconfigurable, so that any member of the LDPC code family described in Section II-C can be accommodated. Address lookup tables can be reconfigured based on the **H** matrix. Processing units can be allocated depending on the column partitions, and the memory size can be adjusted to allow variable code rates.

An alternative version of the sum-product decoder can be implemented using this same architecture. Following the approximation (8), the lookup tables based on Φ are eliminated and the summation in a check node is replaced by comparisons to find the minimum. The approximation results in area savings and the decoder throughput remains the same.

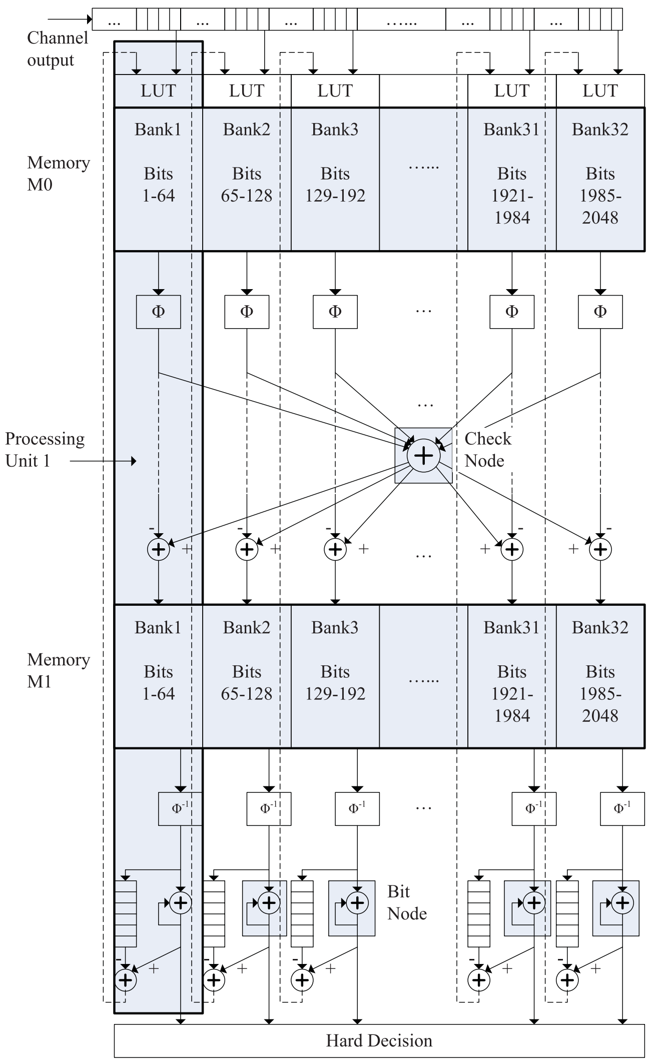


Fig. 2. A parallel-serial architecture of the (2048,1723) RS-LDPC decoder composed of 32 processing units.

E. Decoder Implementation and Emulation Setup

A sum-product decoder for the $(2048, 1723)$ RS-LDPC code has been designed using the Xilinx Virtex-II Pro XC2VP70 FPGA. The decoder is implemented using wordlengths $w = 5, 6, 7$ bits, following Q3.2, Q3.3, Q4.2, and Q5.2 uniform quantization schemes.

Multiple independent AWGN generators have been incorporated on the FPGA using the Xilinx AWGN generator [22]. The probability density function (PDF) of the noise realization deviates within 0.2% from the ideal Gaussian PDF up to 4.8σ [22]. The input to the decoder has to be quantized and clipped so that it can be stored using a limited wordlength. We characterized the binned noise samples produced by the Xilinx noise generator. Even using a crude estimate, we can demonstrate that the true Gaussian error probability curve is within a factor of 3.5 from the results obtained by hardware emulation down to the 10^{-13} level. In our nonlinear finite-wordlength decoding process based emulations we observe that the decoder stalls at very low BERs because of specific patterns of locations in the codeword being subject to noise moderately out in the tail rather than because of noise values in the extreme tails. Thus accuracy of the random number generator in the extreme tail distribution is not of concern in this application, in contrast to what is stated in [23].

Block RAMs on the FPGA record final iterations of soft decisions when decoding fails. An on-chip PowerPC microprocessor controls the decoder, noise generator, and the interface with the memory module. Such a hardware emulation platform allows the characterization of the code and evaluation of practical implementation parameters [3]. Error traces enable the exploration of patterns that cause the decoder to fail.

In a high signal-to-noise ratio (SNR) regime, the majority of the received frames can be decoded in one iteration and the decoder can reach a peak throughput of 240 Mb/s using a 100 MHz clock rate. Hardware emulation of this LDPC decoder extends the BER curve below 10^{-10} within hours. For comparison, an optimized implementation of the same decoder in C provides a peak throughput of only 260 kb/s on an Intel Xeon 2.4 GHz microprocessor.

III. Fixed-Point Quantization Effects and Characterization of Decoding Errors

Both the wordlength and the number of decoding iterations are important design parameters that determine the area, power, and performance of an LDPC decoder. In particular, a short wordlength and a small number of iterations are always desirable in practical implementations. As an illustration, the frame error rate (FER) and the bit error rate versus the signal-to-noise ratio are plotted in Fig. 3(a) showing the effect of iteration number on the performance of a 6-bit (6-b) Q4.2 fixed-point implementation of the $(2048, 1723)$ RS-LDPC sum-product decoder. More iterations result in better performance, although the gain becomes marginal after 50 iterations. So as to minimize the effect of iteration number and to isolate the error events caused by fixed-point implementations, we perform up to 200 iterations. The FER and BER versus SNR curves are shown in Fig. 3(b) for sum-product decoder implementations using Q3.2, Q3.3, Q4.2, and Q5.2 quantization choices.

A. Characterization of Error Events

The definition of absorbing sets has been introduced in our previous work [3], [24], [25]. Absorbing sets provide a valuable characterization of certain types of decoding failure. In order to 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 or factor graph of the parity check matrix $\mathbf{H}$ of a code [26], [27]. For a subset D of V, let $O(D)$ 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 $O(D)$ 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 fewer neighbors in $O(D)$ than in $F \setminus O(D)$.

Related notions have been previously introduced in the literature in the attempt to characterize the behavior of the message-passing decoding algorithms when they do not converge to a codeword, such as stopping sets [28], near-codewords [4], and trapping sets [2]. A fully absorbing set, as defined above, can be understood as a special type of near-codeword or trapping set, one which is stable under the bit-flipping decoding algorithm [1].

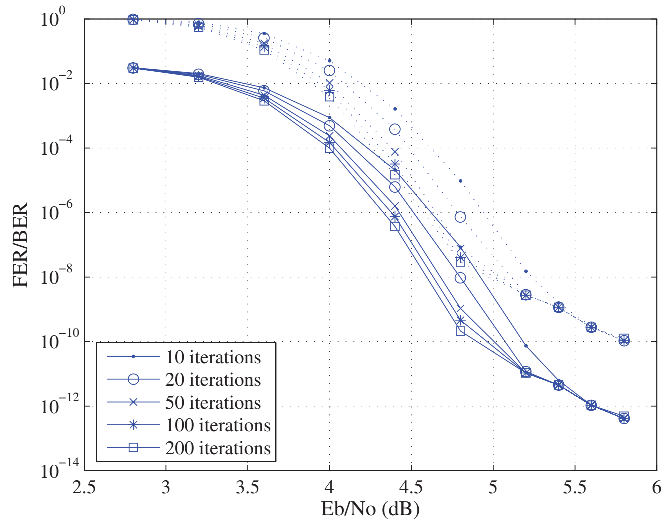
The notion of the absorbing set is being used in this work to resolve the ambiguity in the definitions of objects for describing the error floors. The original definition of the trapping set by Richardson is semi-empirical and decoder-dependent. As a result, three different types of errors could be associated with trapping sets [29]: fixed patterns, oscillatory patterns, and random-like patterns. Subsequent work defined trapping set as a fixed point of the decoder [30]. In contrast, the absorbing set is defined as a combinatorial object, and is decoder independent. Oscillations and random-like errors could be disassociated from absorbing set errors. The combinatorial definition of absorbing set only depends on the structure of the Tanner graph, and therefore the relevant absorbing sets can be systematically enumerated [24], [25]. This exact enumeration of the absorbing sets under iterative decoding can be viewed as being equivalent to identifying the weight enumerator polynomial under maximum likelihood decoding. As such, the absorbing sets of the smallest weight rather than smallest distance codewords determine the performance in the error floor region. In particular, the count of relevant absorbing sets is a key component in developing accurate error floor predictions using importance sampling [31].

Trapping sets are defined in [32] and [33] as any length-n bit vector denoted by a pair (a, b), where a is the Hamming weight of the bit vector and b is the number of unsatisfied checks. An absorbing set could be understood as a special type of such trapping set where each variable node is connected to strictly more satisfied than unsatisfied checks. The satisfied versus unsatisfied notion in the absorbing set definition explains how a fully absorbing set is stable under bit-flipping operations; the implication on practical decoder designs is the focal point of this paper.

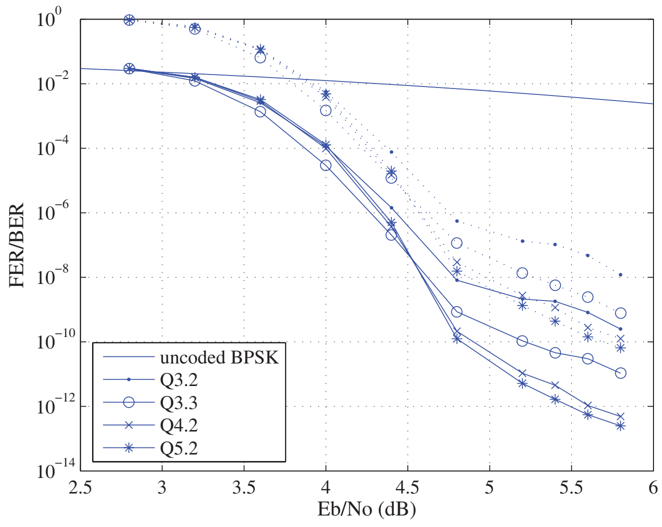
Another related structure is an (a, b) elementary trapping set [32], [33], which is defined as a trapping set for which all check nodes in the induced subgraph have either degree one or two, and there are exactly b degree-one check nodes. Here again, the primary contrast with absorbing sets is the stability of absorbing sets under bit-flipping operations, implied by their definition. The notion of absorbing set can also be refined further by imposing restrictions on vertex and check degree profiles, as done, for instance, later in this paper (see Section IV).

B. Error Analysis

In all the following experiments, an all-zeros codeword is transmitted and the sum-product algorithm is employed to decode the codeword. The final 16 iterations are recorded when the decoder fails to converge to a codeword after 200 iterations. We observe absorbing set errors in cases when the decoder fails to converge and the hard decisions of all bits



(a)



(b)

Fig. 3. FER (dotted lines) and BER (solid lines) performance of (a) the Q4.2 sum-product decoder of the (2048,1723) RS-LDPC code using different number of decoding iterations, and (b) the (2048,1723) RS-LDPC sum-product decoder with Q3.2, Q3.3, Q4.2, and Q5.2 fixed-point quantization using 200 iterations.

TABLE I
ERROR STATISTICS OF (2048,1723) DECODER IMPLEMENTATIONS USING 200 ITERATIONS

SNR (dB)	Errors	5-b (Q3.2)	6-b (Q3.3)	6-b (Q4.2)	7-b (Q5.2)
5.2	Errors collected[1]	142	125	94	46
	(8,8) absorbing sets	18	117	92	45
	Oscillations	116	6	0	0
5.4	Errors collected[1]	56	49	44	40
	(8,8) absorbing sets	8	40	42	37
	Oscillations	47	8	0	0
5.6	Errors collected[1]	51	42	22	33
	(8,8) absorbing sets	8	27	20	30
	Oscillations	41	12	0	0
5.8	Errors collected[1]	52	27	14	20
	(8,8) absorbing sets	6	18	13	16
	Oscillations	44	8	0	0

[1] The total number of frames is not uniform for different SNR levels and quantization choices – more input frames were emulated for higher SNR levels and longer-wordlength quantizations. The number of errors collected is divided by the total number of frames to produce the FER plots in Fig. 3(b).

remain the same for the final iterations. The statistics of the error events are listed in Table I for comparison.

In the 5-b Q3.2 fixed-point implementation, most of the errors in the error floor region display an oscillatory behavior and a small number of errors are caused by $(8,8)$ fully absorbing sets. Examples of the bit error counts illustrating the oscillatory behavior are given in [3]. The oscillatory behavior can be attributed to the dynamics of the message exchange in which a small number of bits propagate incorrect messages through their neighboring unsatisfied checks. These in turn make some of their other neighboring bits admit incorrect values, which are propagated further to more bits. As the number of incorrect bits increases, so do their neighboring checks, which means that after about two steps there is a sufficient number of unsatisfied checks to enforce the correct values. As a result, the total number of incorrect bits decreases again.

The error propagation leading to the oscillatory behavior is related to the quantization choice. Using the Q3.2 uniform quantization, reliable (large-valued) prior LLRs outside the range $[-4, 3.75]$ are clipped, causing underestimation. Variable nodes with underestimated prior LLRs become vulnerable to influence from extrinsic messages. The situation is aggravated by limited resolution (two fractional bits for a resolution of 0.25): the Φ_1 outputs of both reliable (large-valued) and some less reliable (smaller-valued) input messages are both rounded down and the difference between them is lost, resulting in the overestimation of the less reliable extrinsic messages. Underestimated prior LLRs coupled with overestimated less reliable extrinsic messages necessarily encourage error propagation, causing the oscillatory behavior.

A 6-b wordlength allows one more bit for quantization over 5-b. The extra bit can be allocated either to resolution or range increase. An increased resolution reduces the overestimation error of less reliable extrinsic messages and limits error propagation. This is demonstrated by the Q3.3 implementation, where the majority of the errors are due to $(8, 8)$ fully absorbing sets and only a small number of errors are due to oscillations. Alternatively, the extra bit can be allocated for range, as in a Q4.2 implementation. A higher range allows reliable prior LLRs to obtain stronger representations, thus stabilizing the respective variable nodes to prevent oscillations.

The 7-b Q5.2 implementation further improves the error floor performance. All errors collected in Q4.2 and Q5.2 implementations are absorbing errors, and the overwhelming majority of these exhibit the $(8, 8)$ absorbing set structure.

C. Absorbing Set Characterization

As previously discussed, almost all encountered absorbing set errors are of $(8, 8)$ type, all of which are fully absorbing. They share the same structure in which these eight variable nodes participate in a total of twenty-eight checks. Of these, twenty checks are connected with degree-two to the eight variable nodes. Since the girth of the code is at least six [10], these variable node pairs are all different. The remaining eight checks are each connected to a different variable node in the absorbing set. The illustration of such configuration is provided in Fig. 4. Although only a subgraph is drawn, all the $(8, 8)$ sets are indeed fully absorbing sets. For an intuitive explanation of why the failures occur in such a set, suppose that all eight bits in the absorbing set have incorrect values and all other bits have correct values, resulting in all but eight checks being satisfied. These incorrect bits then reinforce each other's incorrect values through the checks they share. In particular, each such bit, along with its incorrect prior, receives five such messages. The correct extrinsic message from its remaining neighboring check cannot overcome this joint effect, and the values remain incorrect. This behavior is also verified experimentally by simulating a floating-point decoder for channel realizations with very noisy inputs in precisely eight bits that constitute an absorbing set, and observing that even the floating-point decoder cannot successfully decode such realizations.

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. This effect is demonstrated in the difference between the performance of the Q4.2 and Q5.2 decoders in the error floor region, whereby in the former case the number of absorbing set failures is higher, leading to a relatively higher error floor.

D. Absorbing Behavior in Finite Number of Decoding Iterations

The number of decoding iterations is usually limited in practice, as it determines the latency and throughput of the system. In the practical high-throughput implementations, the maximum number of iterations for the LDPC decoder is limited to less than ten.

Fig. 3(a) shows that a good performance in the waterfall region can be achieved with as few as ten iterations. The loss in performance in the waterfall region is due to an insufficient number of iterations for the decoding to converge. The 10-iteration BER curve eventually overlaps with the 200-iteration in the error floor region. Analysis of the failures in this region confirms that the $(8, 8)$ fully absorbing set, the dominant cause of error floors in the 200-iteration decoder, causes the 10-iteration decoder to fail as well. This result suggests that in the high SNR region, the absorbing process usually happens very quickly and the absorbing structure emerges in full strength within a small number of decoding iterations. Non-convergent errors, however, become negligible in the error floor region.

IV. Alternative Decoder Implementation and Classification of Absorbing Sets

Finite-wordlength decoders of importance for practical implementations have been studied on a $(5, 47)$-regular $(2209, 1978)$ array-based LDPC code [34]. The class of array-based LDPC codes is known to perform well under iterative decoding [11]. The $\mathbf{H}$ matrix of this code can be partitioned into 5 row groups and 47 column groups of 47×47 permutation submatrices. Note that the regular structure of the $\mathbf{H}$ matrix is well suited for the emulation platform. We perform the following experiments with the wordlength fixed to 6 bits. Unless specified otherwise, we perform a maximum of 200 decoding iterations so as to isolate the quantization effect from the iteration number effect. Using a Q4.2 quantization in a sum-product decoder yields the results shown in Fig. 5(a).

Based on our emulation results, the failures in the error floor region are entirely due to absorbing sets. The statistics of the frequently observed absorbing sets are listed in Table II. The structure of the dominant $(4, 8)$ absorbing set is illustrated in [34]. To facilitate further discussions, we introduce the notation $(p : q)$ to describe the connectivity of a variable node with p connections to satisfied check nodes and q connections to unsatisfied check nodes. In the $(4, 8)$ absorbing set, each variable node in the absorbing set has a $(3 : 2)$ connection. All the other absorbing sets listed in Table II contain variable nodes with $(4 : 1)$ and $(5 : 0)$ connections.

A. Dual Quantization in a Sum-Product Decoder

As the decoder starts to converge, the variable-to-check messages usually grow larger, as their certainty increases. In this regime, the sum-product decoder is essentially operating on the lower right corner of the Φ_1 curve and subsequently on the upper left corner of the Φ_2 curve as highlighted in Fig. 6. We refer to these corners as the operating regions of the Φ_1 and Φ_2 functions. A more accurate representation of extrinsic messages requires more output levels of the Φ_2 function in its operating region, which also necessitates high-resolution inputs to the Φ_2 function. These requirements can be both satisfied if the quantization scheme is designed to have two quantization domains illustrated in Fig. 6. For instance, suppose that Domain A uses a Q4.2 quantization whereas Domain B uses a quantization with a higher resolution, such as a Q1.5 quantization. The 6-b wordlength is preserved to maintain a constant decoder complexity. The functions Φ_1 and Φ_2 separate the two domains. The input to Φ_1 is in a Q4.2 quantization and the output of Φ_1 is in a Q1.5 quantization.

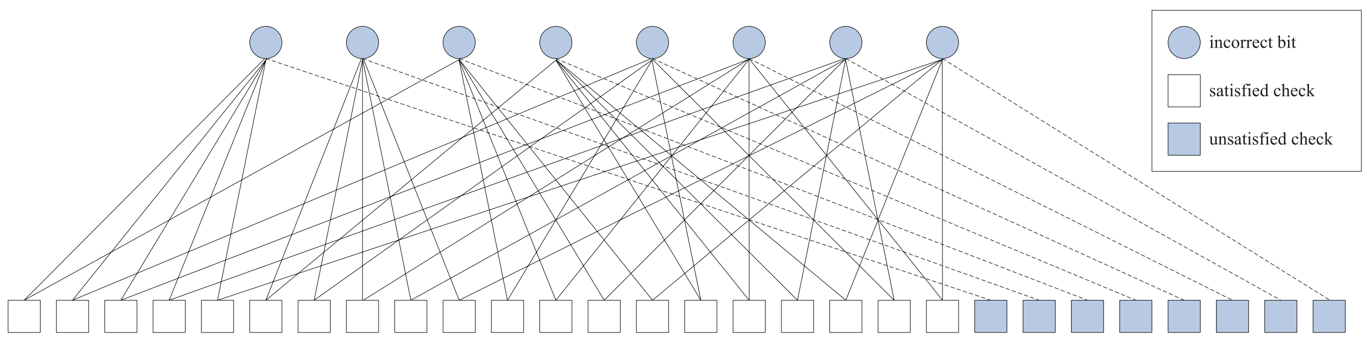


Fig. 4. Illustration of the subgraph induced by the incorrect bits in an (8,8) fully absorbing set.

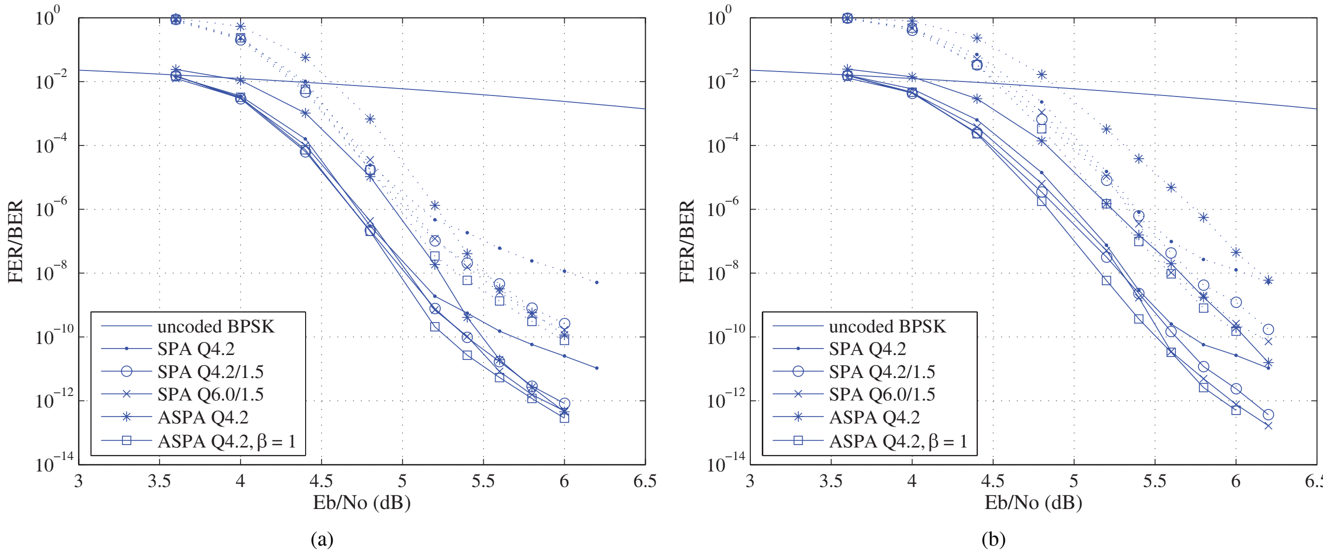


Fig. 5. FER (dotted lines) and BER (solid lines) performance of a (2209,1978) array-based LDPC code using (a) 200 decoding iterations and (b) 10 decoding iterations. (SPA: sum-product decoder, ASPA: approximate sum-product decoder. Q4.2/1.5 and Q6.0/1.5 denote dual quantizations using two domains. Refer to Section IV-A and Fig. 6 for explanations and illustration.

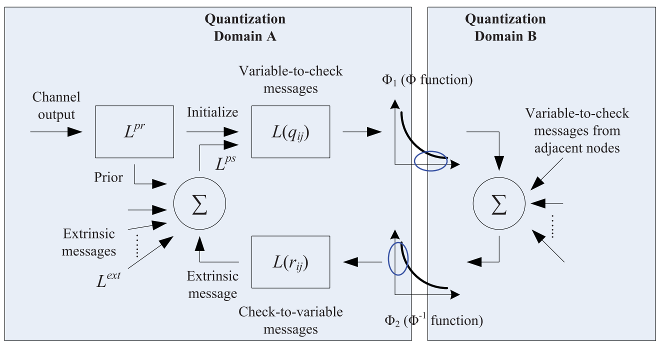

Quantization Domain A
Quantization Domain B
Channel output
L^{pr}
Initialize
Variable-to-check messages
$L(q_{ij})$
L^{ps}
Prior
Extrinsic messages
L^{ext}
Extrinsic message
$L(r_{ij})$
Check-to-variable messages
Φ_1 (Φ function)
Φ_2 (Φ^{-1} function)
Variable-to-check messages from adjacent nodes

Fig. 6. A sum-product decoder with two quantization domains (the operating regions of Φ_1 and Φ_2 functions are circled).

The Φ_2 function assumes the opposite quantization assignment. We refer to this scheme as dual quantization, since the quantization levels are tailored to the operating region within each domain. There is no increase in hardware complexity for implementing this scheme.

Fig. 5(a) shows that the Q4.2/1.5 dual quantization results in better performance than the Q4.2 quantization in both the waterfall and the error floor regions. We attribute the performance advantage of the Q4.2/1.5 dual quantization to more levels in the operating regions of the Φ_1 and Φ_2 functions, which enable a more accurate representation of the extrinsic messages. Reliable extrinsic messages could potentially obtain a stronger representation than the less reliable extrinsic messages, so that the error propagation is limited and the absorbing set errors become less likely.

The $(4, 8)$ and $(5, 9)$ absorbing sets, observed in the Q4.2 quantization, are much less frequent when decoding using the dual quantization scheme, and the error floor is now dominated by $(6, 8)$ and $(8, 6)$ absorbing sets. All of the collected $(6, 8)$ and $(8, 6)$ sets are fully absorbing, with configurations illustrated in [34]. The $(6, 8)$ absorbing set consists of two variable nodes with $(3 : 2)$ connections and four variable nodes with $(4 : 1)$ connections. The $(8, 6)$ absorbing set consists of only variable nodes with $(4 : 1)$ and $(5 : 0)$ connections. Both the $(4 : 1)$ and the $(5 : 0)$ configurations are more stable as absorbing sets than the $(3 : 2)$ configuration, for which reason we consider the $(6, 8)$ and $(8, 6)$ absorbing sets stronger than the $(4, 8)$ absorbing set.

B. Representation of Channel Likelihoods

For practical SNR levels, a Q4.2 quantization scheme does not offer enough range to capture the input signal distribution.

TABLE II
ABSORBING SET PROFILE OF (2209,1978) DECODER IMPLEMENTATIONS

Algorithm[1]& Quantization	SNR (dB)	Errors collected[2]	(4,8)	(5,9)	(6,8)	(7,9)	(8,6)	(8,8)	(9,5)	(10,4)	(10,6)
SPA Q4.2	5.4	185	50	22	34	17	9	13	2		
	5.6	121	39	12	36	9	8	4			
	5.8	104	50	15	11	6		1			
	6.0	50	32	5	5	4					
SPA Q4.2/1.5	5.4	149			16	3	57	9	17	4	3
	5.6	87			21	5	33	8	7		2
	5.8	42	1		6	2	15	8	2	2	2
	6.0	21	2		8		7	2		1	
SPA Q6.0/1.5	5.4	133	1		28	7	16	12	3	1	1
	5.6	66		1	29	5	12	12			
	5.8	38			17	2	7	6	1	1	
	6.0	13			9	2	1				
ASPA Q4.2	5.6	221			2		91	5	36	14	7
	5.8	59				1	30	1	13	3	
	6.0	22					15	1	3	1	
ASPA β=1 Q4.2	5.4	307			6	2	143	17	38	16	12
	5.6	243			6	2	122	13	40	16	9
	5.8	58			1		35	1	8	4	2
	6.0	18			2		9	3	2		1

[1] SPA: sum-product decoder, ASPA: approximate sum-product decoder.
[2] The total number of frames is not uniform for different SNR levels, quantization, and algorithm choices. The number of errors collected is divided by the total number of frames to produce the FER plots in Fig. 5(a).

Moreover, it clips correct priors and incorrect priors disproportionately. By selecting a Q6.0 quantization in Domain A, an increased input range is accepted, which permits correct priors to assume stronger values without being clipped excessively. Variable nodes backed by stronger correct priors cannot be easily attracted to an absorbing set, thus the probability of absorbing set errors is reduced. Statistics in Table II show that the $(6,8)$ and $(8,6)$ sets remain to be dominant. The error floor performance of the Q$6.0/1.5$ dually-quantized decoder improves slightly over the Q$4.2/1.5$ performance. In particular, the Q$6.0/1.5$ dually-quantized decoder performs well in the error floor region even in ten decoding iterations as shown in Fig. 5(b).

C. Approximate Sum-Product Decoding

By using the approximate sum-product algorithm (8) to bypass Φ_1, summation, and Φ_2 altogether, saturation and quantization errors incurred in the log-tanh processing are eliminated. We simplify the Q4.2 sum-product decoder of the $(2209, 1978)$ array-based LDPC code using the approximation (8). The performance of the Q4.2 approximate sum-product decoder is illustrated along with its sum-product counterpart in Fig. 5(a). In the waterfall region, the approximate sum-product decoder incurs nearly 0.2 dB of performance loss due to approximation errors; however, it performs better in the error floor region. The error floor is dominated by $(8,6)$ and $(9,5)$ fully absorbing sets, which both consist of only variable nodes with $(4:1)$ and $(5:0)$ connections. Lower-weight weak absorbing sets $(4,8)$ and $(5,9)$ are eliminated and even instances of $(6,8)$ and $(7,9)$ absorbing sets are reduced.

The lackluster error floor performance of a conventional sum-product decoder compared to an approximate sum-product decoder is largely due to the estimation of the two log-tanh functions. As in the case of the oscillatory behavior, a finite-wordlength quantization of the log-tanh functions causes underestimations of reliable messages and overestimations of unreliable messages. As a result, the reliability information is essentially lost, and soft decoding degenerates to a type of hard-decision decoding where the decisions are based entirely on majority counting. Such a decoding algorithm is susceptible to weak absorbing sets because it disregards the reliability information. In contrast, the approximate sum-product algorithm is better in maintaining the reliability information, so that it is not easily attracted to weak absorbing sets.

The approximate sum-product decoder can be improved using a correction term [5]. We select an offset $\beta = 1$ to optimize the decoder performance. The performance of the offset-corrected decoder is illustrated in Fig. 5(a), where we observe that both the waterfall and the error floor performance are improved. The absorbing set profile shows that the $(8,6)$ and $(9,5)$ fully absorbing sets determine the error floor.

With reduced iteration count, the approximate sum-product decoder incurs almost 0.5 dB of performance loss. However, the loss can be easily compensated after applying the offset correction. In ten iterations, the performance of the offset-corrected approximate sum-product decoder surpasses all the other sum-product decoder implementations as shown in Fig. 5(b).

D. Dominant Absorbing Sets

In previous discussions, we described the configurations of $(4,8)$, $(6,8)$, $(8,6)$, and $(9,5)$ fully absorbing sets. Two simple ways to characterize these sets are by weight and by stability. Everything else being equal, low-weight absorbing sets appear much more frequently when decoding fails. This phenomenon is more pronounced in 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 the $(2209, 1978)$ array-based LDPC code, the $(8, 6)$ and $(9, 5)$ absorbing sets are stronger or more stable, as it is more difficult to escape such absorbing configurations. 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 strong absorbing sets are of greater importance because they dominate the error floors.

In suboptimal decoder implementations where severe message saturations can occur, such as the Q4.2 sum-product implementation, the performance is dictated by low-weight weak absorbing sets, which lead to an elevated error floor. The implementations can be improved to reduce the adverse effects of message saturation and quantization. The error floor performance of better decoder implementations, such as the dually-quantized decoders and the approximate sum-product decoders, are eventually determined by strong absorbing sets.

V. Conclusion

We proposed a parallel-serial, flexible, high-throughput architecture that allows mapping of a family of high-performance LDPC decoders on an emulation platform. We demonstrated that this emulation platform can be used to capture low BER traces down to 10^{-13} for a $(2048, 1723)$ RS-LDPC code and a $(2209, 1978)$ array-based LDPC code.

In addition, we analyzed the error traces, thereby showing that a class of combinatorial structures known as absorbing sets ultimately determines the error floor performance of these LDPC codes. Our study also established the connection between fixed-point quantization choices and the error floor performance of a sum-product decoder: in a low-resolution implementation, the dominant cause of the error floor is oscillatory behavior, which can be corrected with an increase in resolution, or, more effectively, an increase in range, whereas absorbing sets dominate error floors in a high-range implementation and are due to the code construction.

Investigations based on the $(2209, 1978)$ array-based LDPC code allows further isolation of weak from strong absorbing sets. The conventional quantization schemes applied to the sum-product decoder can be suboptimal, thus allowing weak absorbing sets of relatively small size to dominate, thereby leading to an elevated error floor. The proposed dually-quantized sum-product decoder improves the estimation of log-tanh functions, and the approximate sum-product decoder eliminates the log-tanh functions altogether. Both approaches mitigate the effects of weak absorbing sets and lower the error floor even with a small number of decoding iterations.

Results of this work provide insights into a more effective implementation of a high-throughput LDPC decoder for low error rate performance. Intuitions gained from this work enable further characterization of absorbing sets that cause the error floors.

Acknowledgment

The authors wish to acknowledge the contributions of the students, faculty, and sponsors of Berkeley Wireless Research Center and Wireless Foundations. In particular, we would like to thank Pierre-Yves Droz, Chen Chang, Henry Chen, Hayden So, and Imran Haque for help with the BEE2 emulation platform and assistance with the design.

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. Allerton Conf. Commun., Control, Computing*, Monticello, IL, Oct. 2003, pp. 1426-1435.

[3] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. Wainwright, "Investigation of error floors of structured low-density parity-check codes by hardware emulation," in *Proc. IEEE Global Commun. Conf.*, San Francisco, CA, Nov. 2006, pp. 1-6.

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

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

[6] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549-554, Apr. 2005.

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

[8] L. Yang, H. Liu, and R. Shi, "Code construction and FGPA implementation of capacity approaching low error-floor LDPC decoder," *IEEE Trans. Circuits Systems-I: Regular Papers*, vol. 53, no. 4, pp. 892-904, Apr. 2006.

[9] F. Verdier and D. Declercq, "A low-cost parallel scalable FPGA architecture for regular and irregular LDPC decoding," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1215-1223, June 2006.

[10] 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 Commun. Lett.*, vol. 7, no. 7, pp. 317-319, July 2003.

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

[12] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429-445, Mar. 1996.

[13] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673-680, May 1999.

[14] E. Yeo, B. Nikolić, and V. Anantharam, "Iterative decoder architectures," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 132-140, Aug. 2003.

[15] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966-2984, Dec. 2004.

[16] J. Rosenthal and P. O. Vontobel, "Construction of LDPC codes based on Ramanujan graphs and ideas from Margulis," in *Proc. Allerton Conf. Commun., Control, Computing*, Monticello, IL, Oct. 2000, pp. 248-257.

[17] IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1, IEEE Std. 802.16e, Feb. 2006.

[18] IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Std. 802.3an, Sept. 2006.

[19] M. M. Mansour and N. R. Shanbhag, "Low-power VLSI decoder architectures for LDPC codes," in *Proc. International Symp. Low Power Electronics Design*, Monterey, CA, Aug. 2002, pp. 284-289.

[20] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404-412, Mar. 2002.

[21] Xilinx Virtex Series FPGA, Xilinx Corporation. [Online]. Available: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/

[22] Xilinx Additive White Gaussian Noise, Xilinx Corporation. [Online]. Available: http://www.xilinx.com/products/ipcenter/DO-DI-AWGN.htm

[23] D. U. Lee, W. Luk, J. D. Villasenor, and P. Y. K. Cheung, "A Gaussian noise generator for hardware-based simulations," *IEEE Trans. Computers*, vol. 53, no. 12, pp. 1523-1534, Dec. 2004.

[24] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolić, "Analysis of absorbing sets for array-based LDPC codes," in *Proc. IEEE International Conf. Commun.*, Glasgow, UK, June 2007, pp. 6261-6268.

[25] 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 Trans. Inf. Theory*, 2009, to be published.

[26] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520-548, Feb. 2001.

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

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

[29] H. Xiao and A. H. Banihashemi, "Estimation of bit and frame error rates of finite-length low-density parity-check codes on binary symmetric channels," *IEEE Trans. Commun.*, vol. 55, no. 12, pp. 2234-2239, Dec. 2007.

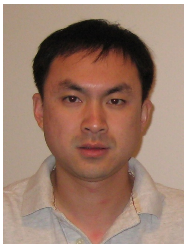
[30] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE International Conf. Commun.*, Istanbul, Turkey, June 2006, pp. 1089-1094.

[31] L. Dolecek, Z. Zhang, M. J. Wainwright, V. Anantharam, and B. Nikolić, "Evaluation of the low frame error rate performance of LDPC codes using importance sampling," in *Proc. IEEE Inform. Theory Workshop*, Lake Tahoe, CA, Sep. 2007, pp. 202-207.

[32] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A general method for finding low error rates of LDPC codes," *IEEE Trans. Inf. Theory*, submitted for publication.

[33] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 39-55, Jan. 2007.

[34] Z. Zhang, L. Dolecek, M. Wainwright, V. Anantharam, and B. Nikolić, "Quantization effects of low-density parity-check codes," in *Proc. IEEE International Conf. Commun.*, Glasgow, UK, June 2007, pp. 6231-6237.



Zhengya Zhang received the B.A.Sc. degree in computer engineering from University of Waterloo, Canada, and the M.S. degree in electrical engineering from University of California, Berkeley. He is currently a Ph.D. candidate in electrical engineering at University of California, Berkeley, where he is a member of the Berkeley Wireless Research Center. His research interest is in the design of signal processing and computation systems which require a spectrum of optimizations from algorithm to architecture and implementation. He is the recipient of the Analog Devices Outstanding Student Designer Award and the Vodafone U.S. Foundation Fellowship for his graduate research. He has held multiple internships with Nvidia and Nortel in the past.

Lara Dolecek is a post-doctoral researcher with the Massachusetts Institute of Technology. She holds a B.S, M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences, as well as an M.A. degree in Statistics, all from the University of California, Berkeley. For her dissertation she received the 2007 David J. Sakrison Memorial Prize for the most outstanding doctoral research in the Department of Electrical Engineering and Computer Sciences at UC Berkeley.

She also received several UC Berkeley-wide awards for her graduate research including the multi-year Eugene Cota-Robles Fellowship and the Dissertation Year Fellowship. Her research interests span information and probability theory, graphical models, combinatorics, statistical algorithms and computational methods with applications to high-performance complex systems for data processing, communication, and storage.

Borivoje Nikolić received the Dipl.Ing. and M.Sc. degrees in electrical engineering from the University of Belgrade, Serbia, in 1992 and 1994, respectively, and the Ph.D. degree from the University of California at Davis in 1999.

He lectured electronics courses at the University of Belgrade from 1992 to 1996. He spent two years with Silicon Systems, Inc., Texas Instruments Storage Products Group, San Jose, CA, working on disk-drive signal processing electronics. In 1999, he joined the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, where he is now a Professor. His research activities include digital and analog integrated circuit design and VLSI implementation of communications and signal processing algorithms. He is co-author of the book *Digital Integrated Circuits: A Design Perspective*, 2nd ed, Prentice-Hall, 2003.

Dr. Nikolić received the NSF CAREER award in 2003, College of Engineering Best Doctoral Dissertation Prize and Anil K. Jain Prize for the Best Doctoral Dissertation in Electrical and Computer Engineering at University of California at Davis in 1999, as well as the City of Belgrade Award for the Best Diploma Thesis in 1992. For work with his students and colleagues he received the Best Paper Award at the ACM/IEEE International Symposium of Low-Power Electronics in 2005, and the 2004 Jack Kilby Award for the Outstanding Student Paper at the IEEE International Solid-State Circuits Conference.

Venkat Anantharam received the B.Tech in Electronics in 1980 from the Indian Institute of Technology, Madras (IIT-M) and the M.A. and C.Phil degrees in Mathematics and the M.S. and Ph.D. degrees in Electrical Engineering in 1983, 1984, 1982 and 1986 respectively, from the University of California at Berkeley (UCB). From 1986 to 1994 he was on the faculty of the School of EE at Cornell University. From 1994 he has been on the faculty of the EECS department at UCB.

Anantharam received the Philips India Medal and the President of India Gold Medal from IIT-M in 1980, and an NSF Presidential Young Investigator award during the period 1988-1993. He a co-recipient of the 1998 Prize Paper award of the IEEE Information Theory Society (with S. Verdu) and a co-recipient of the 2000 Stephen O. Rice Prize Paper award of the IEEE Communications Theory Society (with N. Mckeown and J. Walrand). He received the Distinguished Alumnus Award from IIT-M in 2008. He is a Fellow of the IEEE.

Martin J. Wainwright is currently an assistant professor at University of California at Berkeley, with a joint appointment between the Department of Statistics and the Department of Electrical Engineering and Computer Sciences. He received his Ph.D. degree in Electrical Engineering and Computer Science (EECS) from Massachusetts Institute of Technology (MIT) in 2002. His research interests include statistical signal processing, coding and information theory, statistical machine learning, and high-dimensional statistics. He has been awarded an Alfred P. Sloan Foundation Fellowship, an NSF CAREER Award, the George M. Sprowls Prize for his dissertation research (EECS department, MIT), a Natural Sciences and Engineering Research Council of Canada 1967 Fellowship, the IEEE Signal Processing Society Best Paper Award, and several outstanding conference paper awards.