MEMORY-EFFICIENT TURBO DECODER ARCHITECTURES FOR LDPC CODES

Mohammad M. Mansour and Naresh R. Shanbhag

iCIMS Research Center, CSL
University of Illinois at Urbana-Champaign
Urbana, IL 61801
e-mail: [mmansour, shanbhag]@uiuc.edu

Abstract—**In this paper, we propose a turbo decoding message-passing (TDMP) algorithm to decode regular and irregular low-density parity-check (LDPC) codes. The TDMP algorithm has two main advantages over the commonly employed two-phase message-passing algorithm. First, it exhibits a faster convergence behavior (up to 50% less iterations), and improvement in coding gain (up to an order of magnitude for moderate-to-high SNR and small number of iterations). Second, the corresponding decoder architecture has a significantly reduced memory requirement that amounts to a savings of $(75+25n/\sum$ node-degrees$)\% > 75\%$ for code-length n. A decoder architecture featuring the TDMP algorithm is also presented. Furthermore, we propose a new structure on the parity-check matrix of an LDPC code based on permutation matrices aimed at reducing interconnect complexity and improving decoding throughput. In addition, we construct a wide range of LDPC codes based on Ramanujan graphs which possess this structure.**

1. INTRODUCTION

After the phenomenal success of turbo codes [1], another class of codes exhibiting similar characteristics and performance, called low-density parity-check (LDPC) codes, has gained widespread interest. LDPC codes were introduced by Gallager in 1961 [2] together with the concept of iterative decoding via message passing. However, Gallager's ideas remained largely neglected until the recent rediscoveries made by Mackay-Neal and Wiberg [3, 4]. It was shown [5] that this class of codes is capable of approaching capacity limit at low decoding complexity. LDPC codes were further extended in [5] to include irregular LDPC codes yielding bit-error rates that *theoretically* surpass the performance of the best (turbo) codes known so far.

This paper attempts to promote LDPC codes as *practical* serious competitors to turbo codes at the decoder architecture front as well. All current decoder architectures for LDPC codes such as [6–10] employ the two-phase message-passing (TPMP) algorithm [2]. Due to the random nature of LDPC codes, direct application of the TPMP algorithm results in (1) stringent memory requirements in a serial decoder implementation that limits the applicability of LDPC codes in latency and power sensitive applications, and (2) complex interconnects that limits the amount of inherent parallelism that can be exploited efficiently in a parallel decoder implementation. For this reason, turbo codes still maintain a clear advantage over LDPC codes in mainstream applications (e.g., 3G wireless applications).

In this paper, we depart from the traditional LDPC decoding approach by taking a fresh look at LDPC codes, specifically from the perspective of turbo codes themselves. An LDPC code is viewed as a code concatenated from a number of constituent codes the same way a turbo code is viewed as a code concatenated from a number of convolutional codes. Hence, we propose a *turbo-decoding* message-passing (TDMP) algorithm to decode LDPC codes as an alternative to the TPMP algorithm. The advantages of the proposed TDMP algorithm over the TPMP algorithm are twofold. First, the TDMP algorithm exhibits a faster convergence behavior and an improvement in coding gain over the TPMP algorithm, thus rendering LDPC codes attractive for high-speed applications. Second, it reduces the memory overhead of the TPMP algorithm by more than 75%, thus rendering LDPC codes attractive for low-power applications.

A decoder architecture implementing the proposed algorithm is developed which applies for randomly (in the sense of Gallager's method [2]) or explicitly constructed regular as well as irregular LDPC codes. The architecture can be extended to handle generalized low-density (GLD) codes [11] by simply tailoring the constituent SISO decoders to the syndrome trellises of the sub-codes used. Further, we show that regularity features in the form of permutation matrices simplify the interconnect complexity and hence increase throughput. We show that one method capable of generating exceptional codes, namely the Ramanujan construction method, possesses these features. We also develop a set of transformations that allows for constructing a wider range of regular/irregular LDPC as well as GLD codes.

2. TURBO DECODING OF LOW-DENSITY CODES

2.1. LDPC Codes

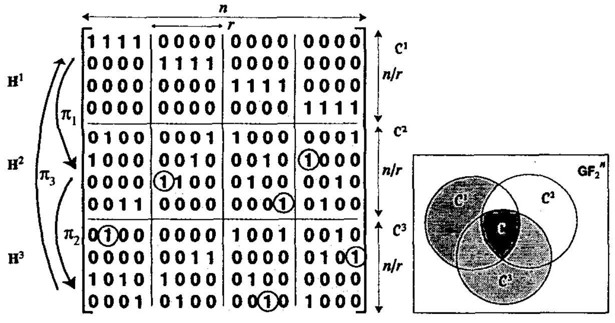
An LDPC code is a linear block code defined by a pseudo-random sparse parity-check matrix $\mathbf{H} = [h_{ij}]_{m\times n}$ and is described by a bipartite graph whose *reduced* adjacency matrix is $\mathbf{H}$. The graph has on one side m *check* nodes $\{c_1, \cdots, c_m\}$ corresponding to the m rows of $\mathbf{H}$, and on the other n *bit* nodes $\{b_1, \cdots, b_n\}$ corresponding to the n columns of $\mathbf{H}$. A bit node b_j is connected to a check node c_i if $h_{ij} = 1$. If all bit nodes have degree c and all check nodes have degree r, the code is called a *regular* (c, r)-LDPC code, otherwise it is called an *irregular* code. LDPC codes are decoded iteratively using Gallager's message-passing (MP) algorithm [2] based on a two-phase schedule that

This work was supported with funds from NSF under grants CCR 99-79381 and CCR 00-73490.

computes iteratively extrinsic probability values associated with each bit node using disjoint parity-check equations that the bit participates in [2]. Each iteration consists of a two-phase round of updates between check and bit nodes according to the connectivity of the underlying graph. Updates in each phase are independent and can be parallelized.

2.2. LDPC Codes Viewed as Concatenated Codes

Gallager [2] constructed ensembles of regular (c, r)-LDPC codes of length n by defining the parity-check matrix $\mathbf{H}_{m \times n}$ of a code $\mathcal{C}$ as a concatenation of c submatrices, each containing a single 1 in each column as shown in Fig. 1(a). The first of these submatrices $\mathbf{H}^1$ having size $\frac{n}{r} \times n$ defines a super-code $\mathcal{C}^1$ as a direct sum of n/r $(r, r-1)$-SPC sub-codes. Note that $\mathcal{C}^1$ satisfies a subset of the parity-check equations of $\mathcal{C}$, and hence $\mathcal{C}$ is a subspace of $\mathcal{C}^1$. The other submatrices $\mathbf{H}^2, \cdots, \mathbf{H}^c$ are pseudo-random permutations of the columns of $\mathbf{H}^1$, each of which define a super-code $\mathcal{C}^j$ on the corresponding subset of the parity-check equations. Hence, $\mathcal{C}$ is the intersection of the super-codes $\mathcal{C}^1, \cdots, \mathcal{C}^c$ as shown in Fig. 1(b). Irregular LDPC codes can similarly be defined by *puncturing* the super-codes. Each punctured super-code $\mathcal{C}^j$ is then a direct sum of n/r $(r_{ij}, r_{ij}-1)$-SPC sub-codes where $r_{ij} \leq r$ and $1 \leq i \leq n/r$ (assuming that puncturing does not result in all-zero rows in $\mathbf{H}$). Figure 1(a) shows an example of a parity-check matrix of a regular $(3, 4)$-LDPC code of length 16 as a concatenation of 3 submatrices. An irregular code can be obtained for example by replacing the encircled **1**'s with **0**'s.



(a) (b)

Fig. 1. (a) A parity-check matrix $\mathbf{H}_{12 \times 16}$ defined as a concatenation of 3 submatrices of size 4×16. Puncturing $\mathcal{C}^2$ and $\mathcal{C}^3$ at the encircled **1**'s results in an irregular code. (b) A code $\mathcal{C}$ viewed as $\mathcal{C}^1 \cap \mathcal{C}^2 \cap \mathcal{C}^3$.

2.3. The Turbo Decoding Message-Passing Algorithm

The previous definition of LDPC codes is reminiscent of parallel concatenated turbo codes [12], and hence it is natural to apply the turbo decoding technique to decode LDPC codes employing constituent SISO decoders to decode each of the super-codes. A similar concept was employed in [11] for decoding GLD codes defined using two super-codes. For each bit, extrinsic probability values are computed given that the bit belongs to $\mathcal{C}^1$. For punctured bits, the extrinsic probabilities are set to their intrinsic values with respect to $\mathcal{C}^1$ (i.e., they are not updated). This extrinsic information is fed as *a priori* information through an interleaver to the SISO decoder working on $\mathcal{C}^2$. Next, the second SISO decoder generates updated extrinsic probability values considering now the unpunctured bits belong to $\mathcal{C}^2$. The process is repeated for all of the remaining super-codes. A single update of messages based on one super-code is called a *sub-iteration*, and a round of updates across all the super-codes constitutes a single decoding iteration. At the final iteration, hard decisions are made based on the *posterior* probability values read (after de-interleaving) from the SISO decoder operating on the last super-code.

The trellis of an unpunctured super-code is a direct sum of n/r syndrome trellises of an $(r, r-1)$ SPC sub-code as shown in Fig. 2, while that of a punctured super-code $\mathcal{C}^j$ is a direct sum of n/r syndrome trellises of $(r_{ij}, r_{ij}-1)$ SPC sub-codes, where $r_{ij} \leq r$ and $1 \leq i \leq n/r$ (assuming no all-zero rows in $\mathbf{H}$). Hence, the SISO decoders of the super-codes consist of independent simple SISO decoders that can operate in parallel on each of the constituent SPC sub-codes. These decoders can either implement Gallager's message update equation [2] or the BCJR algorithm [12]. In [13], it was shown that the BCJR algorithm has a reduced complexity when applied to a 2-state trellis, and is better from an implementation point of view since it is less prone to quantization noise. Moreover, extensions to GLD codes require the use of the BCJR algorithm (or any other *a posteriori* probability decoding algorithm) to compute extrinsic probabilities for general (n_0, k_0) sub-codes instead of simple SPC sub-codes. For SISO-decoder architectures based on the BCJR algorithm, the reader is referred to [14].

An important advantage of the turbo decoding algorithm is that it does not require the saving of multiple check-to-bit messages as is the case for the two-phase MP algorithm; only the most recently generated check messages by a super-code are retained as the bit messages of the succeeding super-code in the chain. This results in faster convergence as demonstrated through simulations in the last Section since new messages are computed based on updated values from each sub-iteration rather than old messages from the previous iteration. At the same SNR, the algorithm requires $20\% - 50\%$ fewer iterations to converge, and can achieve up to an order of magnitude improvement in coding gain for a small number of iterations and moderate-to-high SNR's. This is obviously favorable from an implementation point of view since it improves decoding latency.

3. DECODER ARCHITECTURE

This section presents an architecture implementing the turbo decoding algorithm for (ir-)regular LDPC and GLD codes. Besides its improved decoding latency due to its faster convergence characteristics, the turbo decoding algorithm has significantly reduced memory requirements in a decoder implementation. Unlike the TPMP algorithm where all intermediate messages between bit and check nodes in the underlying bipartite graph must be saved, only the partial bit messages that result from decoding each super-code need to be saved. In fact, for c concatenated super-codes, only the partial bit messages from

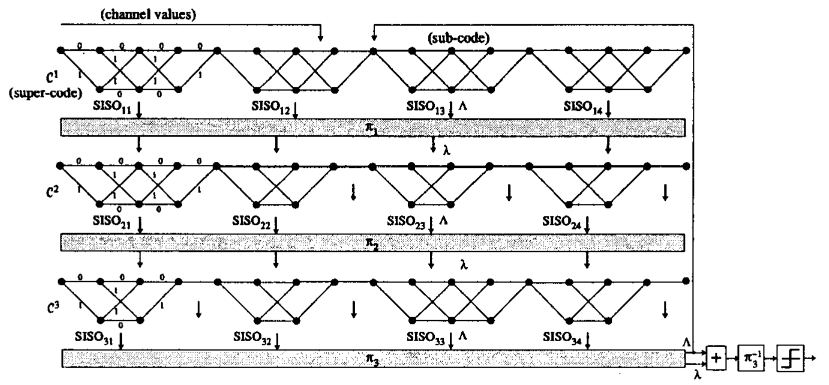


Fig. 2. The turbo decoding algorithm applied to the punctured LDPC code in Fig. 1(a) with (3,2) and (4,3)-SPC sub-codes having simple 2-state trellises of length 3 and 4.

$(c-1)$ of the super-codes need to be saved. For irregular codes, storage for messages corresponding only to unpunctured bits is needed, or $(c_j - 1)$ messages where c_j is the node-degree of the jth bit. In essence, since the overall code is decoded by decoding the super-codes in tandem in one iteration ($= c$ sub-iterations), newly generated bit messages from preceding super-codes are *directly* used to generate bit messages of succeeding super-codes on the same iteration. This eliminates the constraint of saving multiple bit and check messages.

Figure 3(a) shows an architecture implementing the turbo decoding algorithm for low-density codes. The same architecture can be used to decode both LDPC as well as GLD codes where only the constituent SISO decoders of the super-codes differ. The bit messages are stored in n circular buffers each having $(c_j - 1)$ stages. The bits can be partitioned according to their node-degrees $d_1, \cdots, d_k$ so that the ith partition contains n_i bits of degree d_i. The circular buffers of the ith partition form a $(d_i - 1) \times n_i$ array corresponding to $(d_i - 1)$ of the c super-codes as shown in Fig. 3. The n channel reliability values are stored in the buffers MEM_λ. A set of L SISO decoders operate in parallel on a row of the array at any given sub-iteration through the (de-)multiplexers. The parameter L is a VLSI design parameter which is a tradeoff between throughput and interconnect complexity of the (de-)multiplexers.

At the start of every new iteration, messages corresponding to the *second* super-code occupy the tails of the circular buffers, and those for the *last* super-code occupy the heads. During the first sub-iteration corresponding to updates from super-code 1, appropriate messages are read from the buffers through the MUX's and then accumulated to form the total reliability values available with respect to super-code 1 (note that the resulting sum does not include the messages previously generated by super-code 1 itself). Each read operation from the buffers rotates the buffers by $(c_j - 2)$ positions so that the messages to be updated by the SISO decoders are available at the heads of the buffers. The SISO decoders generate the corresponding updated messages for super-code 1 which overwrite old messages corresponding to super-code 2 (at the heads of the buffers) since these are not used in computing new messages for super-code 2 in the next sub-iteration. A sub-iteration terminates when a complete row of messages in the partitions accessed by the SISO decoder is updated. The process is repeated in the remaining $(c-1)$ sub-iterations. The sequence of buffer updates for $c = 4$ is shown in Fig. 3(b) for both unpunctured and punctured super-codes.

The total storage required for the messages is $\sum_{j=1}^{n}(c_j - 1)$ compared to $4\sum_{j=1}^{n} c_j$ for the two-phase MP algorithm, which corresponds to a savings of $(75 + 25n/\sum \text{node-degrees})\% > 75\%$. In the next Section, we consider techniques aimed at improving throughput and reducing complexity of the (de-) multiplexers as well by imposing structural regularity on the parity-check matrix $\mathbf{H}$ defining an LDPC code.

4. LDPC CODES FROM RAMANUJAN GRAPHS

In this section, we develop LDPC codes having special structural regularity on their parity-check matrix that reduces interconnect complexity and improve throughput of the decoder architecture in Fig. 3(a). One particularly favorable regularity feature is the decomposition of $\mathbf{H}$ into *permutation* matrices. A square matrix $\mathbf{P}_{k\times k}$ is called a permutation matrix if each row and each column of $\mathbf{P}$ has exactly one non-zero entry, or equivalently, a permutation of the columns (or rows) of the identity matrix $\mathbf{I}_{k\times k}$. The presence of permutation matrices simplifies the interconnection network and the control logic in Fig. 3.

Short length LDPC codes defined from permutation matrices were introduced in [15, 16]. It was shown that their performance matches and even exceeds those of comparable randomly constructed LDPC codes. In [13], we proposed a construction of the permutation matrices based on cyclotomic cosets (where a specific choice of cosets gives the construction in [15]), and showed the advantages of permutations matrices in simplifying interconnect network and improving throughput. Referring back to the decomposition shown in Fig. 1(a), the submatrices $\mathbf{H}^1, \cdots, \mathbf{H}^c$ of the super-codes would be composed of permutation matrices under this method. However, in general such construction method does not yield impressive codes for long code lengths. One notable construction method capable of generating asymptotically good codes is the explicit algebraic

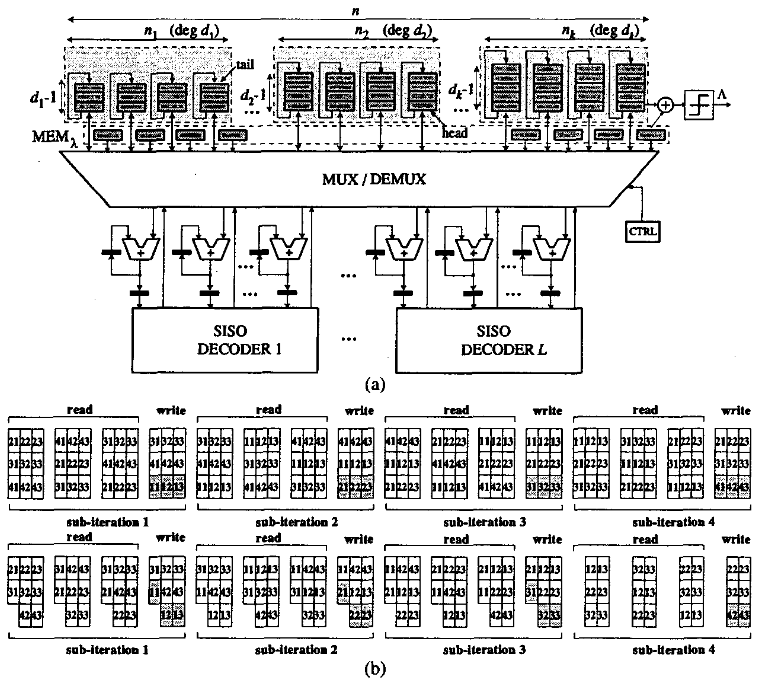


Fig. 3. (a) Decoder architecture implementing the turbo decoding algorithm. (b) Sequence of updates of the lth circular buffer for unpunctured (top) and punctured (bottom) super-codes for $c = 4$.

construction method based on Ramanujan graphs. It will be shown that the parity-check matrix of a Ramanujan-based LDPC code has the desired regularity property in which the submatrices $\mathbf{H}^1, \cdots, \mathbf{H}^c$ are composed of permutation matrices and all-zero matrices.

Ramanujan Construction Method. A bipartite graph can be used to define an LDPC code. Of particular interest are those graphs that have large *girth* (length of shortest cycle) and are good *expanders* (informally, propagate information quickly among the nodes in the graph). Independently, Margulis and Lubotzky-Philips-Sarnak (LPS) [17, 18] found a special class of graphs that posses, among others, the above two extremal properties of large girth and expanding coefficient. The class of graphs defined by LPS is constructed algebraically based on two distinct odd primes p and q and denoted by $X_{\text{RG}}(q,p)$. The parameter p defines the node-degrees of the graph, and q determines the number of vertices in the graph. If p is a quadratic non-residue modulo q, the resulting graph is a biregular bipartite graph having $\frac{q^3-q}{2}$ vertices in each partition and uniform node-degrees of $(p+1)$. These graphs are called Ramanujan because all non-trivial eigenvalues of their reduced adjacency matrix $\mathbf{A}_{\text{RG}}(q,p)$ have magnitude at most $2\sqrt{p}$.

LDPC codes constructed from Ramanujan graphs were studied in [19] following the ideas of Margulis [17] and in [20]. In [21], the construction method of Ramanujan graphs was generalized to the case $q = m$ is composite allowing the construction of a wider range of LDPC codes with flexible parameters. It can be shown [22] that under certain conditions on m, the resulting Ramanujan graph is a biregular bipartite graph $X_{\text{RG}}(m,p)$ having $|X_{\text{RG}}| = \frac{m^3}{2^{h_2}} \prod_{i=1}^{h_1+h_2} (1 - \frac{1}{q_i^2})$ vertices and uniform node-degrees of $(p+1)$. The parameters h_1, h_2, and q_i are defined based on the prime factorization of m into $\prod_{j=1}^{h_1} q_j^{r_j} \cdot \prod_{j=h_1+1}^{h_1+h_2} q_j^{r_j}$ where $\{q_1, \cdots, q_{h_1}\}$ are the prime factors for which p is a quadratic non-residue and $\{q_{h_1+1}, \cdots, q_{h_1+h_2}\}$ are those for which p is a quadratic residue.

Theorem 1 [22] *The reduced adjacency matrix $\mathbf{A}_{\text{RG}}(m,p)$ of a Ramanujan graph $X_{\text{RG}}(m,p)$ can be partitioned into an $N \times N$ array of square matrices $W_{S\times S}$ where $N = m\prod_{q|m}(1+\frac{1}{q})$ and $S = \frac{m^2}{2}\prod_{q|m}(1-\frac{1}{q})$. Each W is either $\mathbf{0}_{S\times S}$ or the sum of one or more non-overlapping permutation matrices.*

Example 2 For $m = 9$ and $p = 5$, the Ramanujan graph $X_{\text{RG}}(9,5)$ has 720 vertices and node-degree 6. Its adjacency matrix $\mathbf{A}_{\text{RG}}(9,5)$ has size 340, and can be partitioned into a 12×12 array of matrices each of which is either a permutation matrix $\mathbf{P}_{27\times27}$ or $\mathbf{0}_{27\times27}$ as shown in Fig. 4.

The next step is to transform the bipartite Ramanujan graph into a factor graph $X_{\text{FG}}(m,p)$ that defines an LDPC code, or alternatively, to transform the square matrix $\mathbf{A}_{\text{RG}}(m,p)$ into a rectangular parity-check matrix $\mathbf{H}_{\text{FG}}(m,p)$ as shown in Fig. 5. A set of possible transformations of the columns of $\mathbf{A}_{\text{RG}}(m,p)$ is shown in the middle row in Fig. 6, and the bottom row shows the corresponding transformations with respect to a node x in the right partition of X_{RG} and its neighbors $a, b, \cdots, f$ on the left.

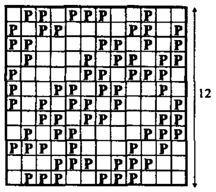


Fig. 4. $\mathbf{A}_{\mathrm{RG}}(9,5)$.

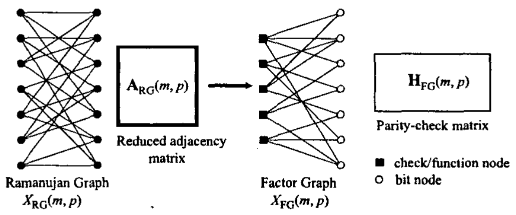


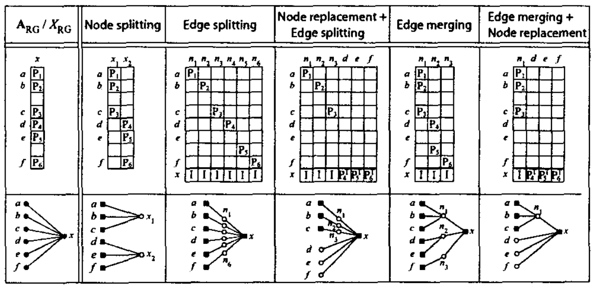
Fig. 5. A factor graph from a Ramanujan graph.

The first transformation, *node splitting*, splits x into two nodes x_1 and x_2 which become bit nodes in X_{FG}, while its neighbor nodes $a, b, \cdots, f$ become check nodes in X_{FG}. In the *edge splitting* transformation, the edges connecting x to $a, \cdots, f$ are split and new nodes $n_1, n_2, \cdots, n_6$ are inserted. The nodes $x, a, b, \cdots, f$ become check nodes while $n_1, n_2, \cdots, n_6$ become bit nodes. All bit nodes have degree 2 and hence this case is suitable for GLD codes with sub-codes having minimum distance at least 3. In the third transformation, a subset of the neighbors, say d, e, f, are *replaced* with bit nodes and the edge splitting transformation is applied to remaining nodes a, b, c and x. Alternatively, edges can be merged together so that they become incident on new nodes which are then taken as bit nodes such as n_1, n_2, n_3 in the *edge merging* transformation. Finally, the fifth transformation is a hybrid transformation of edge merging and node replacement. The node splitting transformation corresponds to Type 1 and 4 replacements in [23] while edge replacement corresponds to Type 3 and 4 replacements. The parity-check matrices resulting from the application of the node splitting and edge merging transformations on $\mathbf{A}_{\mathrm{RG}}(9,5)$ in Fig. 4 are shown in Fig. 7.

In general, the number of super-codes is a multiple of N (the number of matrices per matrix column in $\mathbf{A}_{\mathrm{RG}}$), say $\alpha N \in \mathbb{Z}$. The (de-)multiplexers of the decoder architecture in Fig. 3(a) reduce to a set of 1-to-S (de-)multiplexers and the control logic simply implements the corresponding permutations. Each super-code can be decoded in 1 sub-iteration using S parallel SISO decoders, and hence one iteration completes in $\alpha N \tau_{\mathrm{tot}}$ where τ_{tot} is the delay of the decoders and the (de-)multiplexers. For I decoding iterations per frame and clock frequency f, the throughput is $nf/(\alpha N I \tau_{\mathrm{tot}})$ bits/sec.

5. SIMULATION RESULTS

The performance of the turbo decoding algorithm is compared with the TPMP algorithm by simulating LDPC codes constructed from Ramanujan graphs using the proposed the transformations over an AWGN channel with BPSK modulation. The parameters of the designed codes are CODE 1: ($m = 15, p = 7, n = 1440$, rate $= \frac{1}{2}$) using node splitting, CODE 2: ($m = 15, p = 7, n = 2880$, rate $= \frac{1}{2}$) using edge merging, CODE 3: ($m = 21, p = 11, n = 6048$, rate $= \frac{1}{3}$) using edge merging, and CODE 4: ($m = 33, p = 7, n = 15\,840$, rate $= 0.504$) using node splitting. Figure 8(a) compares the performance of CODE 1 using the turbo decoding algorithm (left) and the two-phase MP algorithm (middle), and shows the number of iterations required for convergence (right) as a function of SNR. Figure 8(b) shows the results for CODE 2. The figures demonstrate that the proposed algorithm converges faster than the commonly employed two-phase MP algorithm (up to 50% less iterations), and yields higher coding at moderate-to-high SNR and small number of iterations (up to 2 orders of magnitude reduction in BER). Finally, Fig. 8(c) demonstrates that the Ramanujan construction method of LDPC codes coupled with the proposed transformations is capable of generating good LDPC codes with large lengths such as CODE 3 and 4.



$\mathbf{A}_{\mathrm{RG}}/X_{\mathrm{RG}}$
Node splitting
Edge splitting
Node replacement + Edge splitting
Edge merging
Edge merging + Node replacement

Fig. 6. Ramanujan to factor graph transformations.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," in *IEEE Int. Conf. on Communications*, 1993, pp. 1064–1070.

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

[3] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. on Info. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[4] N. Wiberg, *Codes and decoding on general graphs*, Ph.D. thesis, Linköping University, Sweden, 1996.

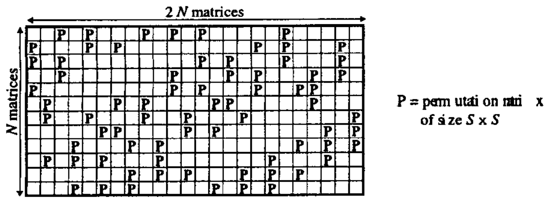
[5] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Info. Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001.

[6] C. Howland and A. Blanksby, "Parallel decoding architectures for low density parity check codes," in *Proc. of 2001 IEEE Int. Sympos. on Circuits and Systems, Sydney*, May 2001, pp. 742–745.

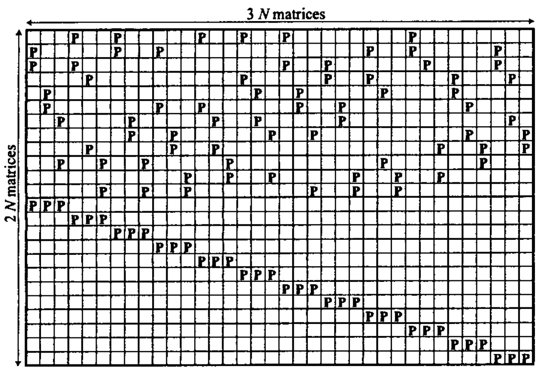
[7] E. Yeo et al., "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. on Magnetics*, vol. 37, no. 2, pp. 748–755, Mar. 2001.

[8] T. Zhang and K. K. Parhi, "VLSI implementation-oriented (3,k)-regular low-density parity-check codes," in *SiPS 2001, Antwerp, Belgium*, Sept. 2001, pp. 25–36.

[9] E. Yeo et al., "High Throughput Low-Desity Parity-Check Decoder Architectures," in *Proc. IEEE Globecom2001, San Antonio, TX*, Nov 2001, pp. 3019–3024.

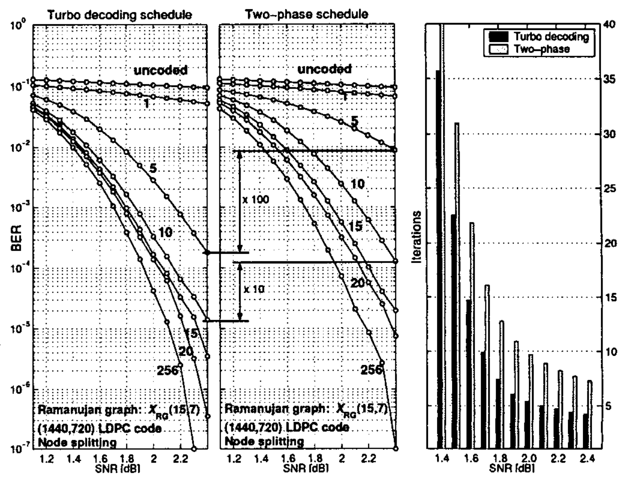


(a)

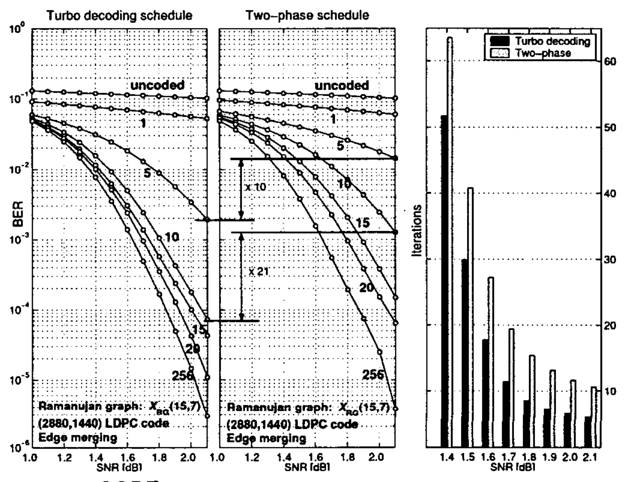


(b)

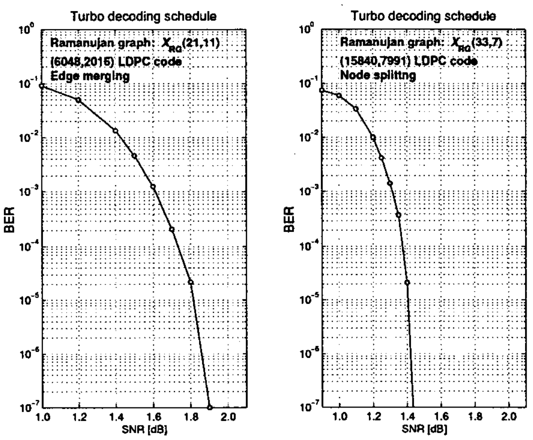
Fig. 7. $\mathbf{H}_{\mathrm{FG}}(9, 5)$ obtained using (a) node splitting and (b) edge merging transformations.



(a) CODE 1: turbo decoding vs. the two-phase MP algorithm.



(b) CODE 2: turbo decoding vs. the two-phase MP algorithm.



(c) CODE 3 (left) and CODE 4 (right).

Fig. 8. BER simulation results.

[10] G. Al-Rawi and J. Cioffi, "A highly efficient domain-programmable parallel architecture for LDPCC decoding," in *ITCC 2001*, April 2001, pp. 569–577.

[11] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (tanner) codes," in *Proceedings of ICC'99*, Vancouver, June 1999, pp. 441–445.

[12] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. on I.T.*, pp. 284–287, Mar. 1974.

[13] M. M. Mansour and N. R. Shanbhag, "Low-power VLSI decoder architectures for LDPC codes," *ISLPED 2002 (to appear)*, Aug. 2002.

[14] M. M. Mansour and N. R. Shanbhag, "Turbo decoder architectures for low-density parity-check codes," *GlobeCOM 2002 (to appear)*, Nov. 2002.

[15] D. Sridhara, R. M. Tanner, and T. E. Fuja, "Low Density Parity Check Codes from Permutation Matrices," in *2001 Conference on Information Sciences and Systems*, Mar. 2001.

[16] J. L. Fan, "Array Codes as Low-Density Parity Check Codes," in *2nd International Symposium on Turbo Codes and Related Topics (Brest, France)*, Sep. 2000.

[17] G. A. Margulis, "Explicit Constructions of Graphs Without Short Cycles and Low Density Codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.

[18] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan Graphs," *Combinatorica*, vol. 8, no. 3, pp. 261–277, 1988.

[19] J. Rosenthal and P. O. Vontobel, "Constructions of LDPC Codes Using Ramanujan Graphs and Ideas from Margulis," in *Proc. of the 38-th Allerton Conference on Communication, Control, and Computing*, 2000, pp. 248–257.

[20] J. Lafferty and D. Rockmore, "Codes and Iterative Decoding on Algeraic Expanders," in *ISIT 2000, Hawaii, U.S.A*, Nov. 2000.

[21] M. M. Mansour and N. R. Shanbhag, "Construction of LDPC Codes from Ramanujan Graphs," in *Conference on Info. Sciences and Systems 2002, Princeton University*, Mar. 2002.

[22] M. M. Mansour and N. R. Shanbhag, "Algebraic Construction of Ramanujan Graphs," CSL technical report, UIUC.

[23] P. O. Vontobel and H.-A. Loeliger, "Irregular Codes from Regular Graphs," *ISIT 2002* (to appear), 2002.