High-Throughput VLSI Implementations of Iterative Decoders and Related Code Construction Problems

Vijay Nagarajan$^+$, Nikhil Jayakumar^o, Sunil Khatri^o, Olgica Milenkovic$^+$

$^+$*Electrical and Computer Engineering Department, University of Colorado Boulder, Colorado 80309*

^{o}*Department of Electrical Engineering, Texas A&M University, College Station, TX 77843*

e-mail: $^+${ nagaraja, milenkov }@colorado.edu, ^{o}{ nikhil, sunil }@ee.tamu.edu

Abstract: ***In this paper, an efficient, fully-parallel Network of Programmable Logic Array (NPLA)-based realization of iterative decoders for structured LDPC codes is presented. The LDPC codes are developed in tandem with the underlying VLSI implementation technique, without compromising chip design constraints. The codes are based on a novel modification of array codes. This design methodology results in reduced routing congestion, a major problem in prior approaches. The operating power, delay and chip-size of the circuits are estimated, indicating that this implementation significantly outperforms presently used standard-cell based architectures. The described LDPC design method can accommodate widely different requirements, such as those arising from recording and wireless channel applications.***

1. Introduction

One of the most promising capacity-approaching error-control techniques in communication theory is coding with Low-Density Parity-Check (LDPC) matrices, coupled with decoding on a graphical representation of the code. Long, random-like LDPC codes offer the best quality error-control performance for a wide range of applications. In addition to their excellent performance, LDPC codes are endowed with an inherently parallel, linear time-complexity decoding algorithm. This makes them amenable for use with parallel VLSI architectures. The primary limiting factor of most known parallel designs is wireability, which arises due to the complexity of the graph interconnection rules of random-like LDPC codes. Additional problems arise from the fact that random-like LDPC codes also require large block sizes for good error correction performance, leading to prohibitively large chip sizes. Despite these bottlenecks, there were several attempts to develop high throughput implementations and implementation-oriented code constructions [1,5,9,10]. The drawback of most of the proposed techniques is that the code-design and VLSI implementation issues are considered in a somewhat decoupled manner, resulting in increased chip dimension and reduced data throughputs. As an example, the standard-cell based approach adopted in [1] has a die area of 7.5 mm x 7 mm for a rate 1/2 code of relatively short length; the design strategy followed in that and some other reports is based on choosing some known random or structured coding scheme, and developing a good implementation for it [1,5,9,10]. Strategies of this type most often rely on utilizing optimization techniques that fail to be efficient for code lengths beyond several hundreds. Besides, they do not address the need of high throughput, low-to-moderate redundancy codecs used in recording and wireless communication systems. Furthermore, for the applications mentioned above, the decoder is usually only one part of a significantly larger system incorporating other blocks such as channel detectors. Hence, it becomes even more important to make the coder/decoder designs as efficient as possible.

In this paper we address the problem of LDPC code construction and VLSI implementation from a different and significantly broader perspective. The crux of our approach is that VLSI implementation-aware code design can lead to an exceptional increase in data throughputs and overall efficiency by means of a careful choice of VLSI implementation and circuit design techniques. In this context, joint optimization of code-related and hardware imposed code-constraints is performed. The first set of constraints includes characteristics such as large girth and large minimum distance of the codes; the second set of constraints is related to VLSI issues such as routing congestion, cross-talk minimization, uniform processing delay in one iteration, power conservation, and chip size reduction. The approach proposed in this work is to devise a fully parallel implementation based on the use of an NPLA. Implementing a circuit using medium sized PLAs was shown to result in fast and area efficient systems [2,3]. The Check/Variable nodes in an LDPC decoder can be decomposed into such a network resulting in the ability to implement fully parallel LDPC decoders in an efficient manner. This fully parallel implementation also eliminates the need for storing the code description in memory - the code structure is implicit in the wiring of the chip itself. The presented results indicate that the PLA-based designs have a very small chip size and low power consumption even for codes of large rate and that they offer a high level of operational flexibility. The system throughput is limited by the rate at which the integrated circuit (IC) is able to read in serial data (approximately 10Gbps), but it can accommodate orders of magnitude larger serial data rates.

The paper is organized as follows. In Section 2, we briefly describe the LDPC decoder and the VLSI implementation issues. Section 3 discusses problems related to the design of structured LDPC decoder ICs and introduces the technical details needed for describing the proposed VLSI architecture. Section 4 contains an overview of the proposed layout while Section 5 contains the description of the structure of the

LDPC codes supporting the proposed layout. The power, area and throughput estimates are presented in section 6. Section 7 concludes the paper with directions for future research.

2. LDPC Codes: Implementation Bottlenecks

A consequence of the graphical representation of LDPC codes is that they can be efficiently decoded in an iterative fashion. In the bipartite graph of a rate 1-*m*/*n* code, the *m* rows of the parity-check matrix *H* represent check nodes ("right nodes"), while its *n* columns represent variable nodes ("left node"). For the clarity of exposition, throughout the rest of the paper only regular codes with column and row weight d_v and d_c, respectively, will be considered. All the proposed ideas and design methods can be modified and extended to also accomodate irregular codes.

Based on the well-known iterative check and variable node update equations [4], and the structure of the code-graph, the implementation issues for the decoder can be seen to be:

- *Large wiring overhead and routing congestion.*
- *Approximate computations performed at check nodes, involving tanh and arctanh functions.*
- *Finite precision arithmetic involved in the hardware implementation.*
- *Problems related to the throughput of the system.*

Currently known implementations fail to provide solutions for one or more of these problems. Ideally, in order to address the first issue, code structures that bound the worst-case wire length are needed. None of the available approaches takes this constraint into account. Furthermore, the resulting design should be area and delay efficient. As will be shown later, for medium sized functions, standard-cell implementations used so far have large area and delays. Thereby, current implementations fail to cater to the needs of modern communications/storage systems, which constitute the major commercial markets for LDPC codes.

3. The Approach: Structure and Full Parallelism

For the purpose of implementing a fully parallel LDPC decoding system as a solution to the above problems, we propose to use extremely fast and area-efficient networks of Programmable Logic Arrays (PLAs) [2,3]. The major components of the proposed system are:

- Full parallelism with the code "embedded" in the wiring.
- Efficient implementation with PLAs.
- Unified approach in tackling code design and VLSI implementation constraints.

This approach yields throughputs of the order of 100s of *Gbps*. Since such a system can process information at any incoming rate possible in present schemes, it can fit into most modern recording and wireless systems. The LDPC codes are tailor-made to meet performance as well as VLSI placement and routing constraints. Such an approach yields to an overall solution of the problem that demonstrates significant improvement over prior attempts to implement LDPC codecs in VLSI. The parallel nature of the iterative decoding graph can be directly implemented in hardware. Since each of the variable and check nodes makes use of information available from their counterparts only from the previous cycle, it is possible to let these units operate in parallel and complete their operations in one clock cycle. But the main challenge in this implementation is to reduce the wiring complexity. This problem is solved at the code design level itself. The LDPC codes are hardwired into the chip and have a structure that results in small wiring overhead and also helps avoid storing the code parity check matrix.

A. Variable Node Hardware Architecture

The outgoing information through any edge of a variable node is the sum of the channel information and the information coming into the variable node from all other edges. Assuming *5-bit* quantized messages both from the channel and the check nodes, a total of $log(d_v+1) +1$ stages (levels) of two-input adders is needed to perform the variable operations. For this purpose, pre-charged *Manchester adders* described in [8] are used. At the beginning of the evaluate-period of a clock cycle, the messages from the previous iterations are used to perform a series of additions. The results of these additions are latched and sent as inputs to the check nodes during the next clock cycle. The sign of the sum represents the current estimate of the decoded bit. Though it is possible to increase the throughput by stopping the iterative process for the block by checking for the correct parity, the proposed architecture does not incorporate this feature. Figure 1 shows the proposed variable node architecture.

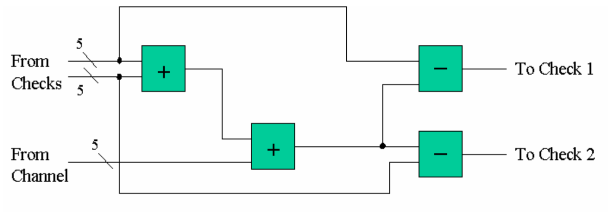


Figure 1: Variable node architecture (d_v=2)

B. Check Node Hardware Architecture

At the check nodes, two operations are performed, namely the parity updates and the reliability updates. Since the parity update operation implementation has been dealt with in [1] and has a very small influence on the chip area and power overhead, it is not described in this paper. The reliability updates are performed in the log-likelihood domain in order to avoid multiplication and division operations. The system blocks are required to:

a) Perform *logtanh* operation on each incoming edge;
b) Add all such *logtanh* values on a check node;
c) Subtract the incoming edge on each edge;
d) Perform an inverse *logtanh* operation on these subtracted messages on each of the edges to obtain the "outgoing" information from the variable nodes at the end of an iteration.

Figure 2 shows the reliability update architecture at the check nodes for the case $d_c=3$. Finite precision arithmetic is used to develop a PLA-based look-up for the *logtanh* and *logarctanh* operations. For the 5-bit precision that is utilized, the PLAs implement the *logtanh* and *logarctanh* operations as lookup tables, with 5 inputs, 5 outputs and 32 rows per PLA.

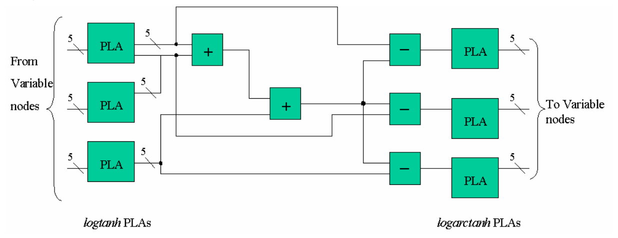


Figure 2: Architecture for Reliability update in check node

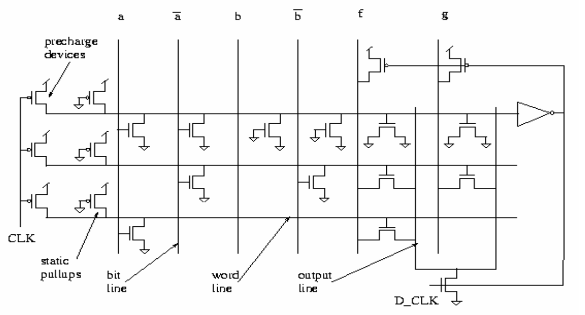


Figure 3: Schematic of the PLA core

A pre-charged NOR-NOR style of PLAs [2,3] is used for the proposed design. The schematic view of the PLA core is shown in Figure 3. In a pre-charged NOR-NOR PLA, each word-line of the PLA switches from high to low at the end of any computation, if it switches at all. As a result, there is no delay deterioration effect due to crosstalk with neighboring word-lines. One maximally loaded word-line is designed to switch low in the evaluate-phase of every clock. It effectively generates a delayed clock, D_CLK, which holds up the evaluation of the other word-lines until they have switched to their final values.

Example	PLA-based		Standard Cell		Ratios	
	D	A	D	A	D	A
cmb	160.3	53.3k	300	159.8k	0.534	0.334
Cu	189.1	69.5k	420	186.5k	0.450	0.373
X2	164.8	45.3k	290	136.8k	0.568	0.331
Z4ml	200.5	95.2k	575	118.3k	0.349	0.805

Table 1: Comparison of standard-cell and PLA-based implementation styles

Figure 4 shows the layout of the PLA core implemented using two metal layers. Delay and area comparisons for PLA-based and standard-cell based implementations for four test examples are shown in Table 1. For each layout style, *D* denotes the delay in picoseconds and *A* denotes the layout area of the resulting implementation in square grids. Note that for the standard cell layout style, the *D* values are the maximum values obtained after simulating about 20 input vectors. For the PLA-based implementations, *D* is the same irrespective of the input vectors since it is determined from the maximally loaded lines. Wire capacitances in the standard cell implementation, which would only increase its delay, are not accounted for. In the case of the PLA layout style, however, the delay numbers are worst-case values. Despite this, the PLA layout style shows impressive improvements over the standard cell layout style. The reason why PLAs result in very favorable area and delay characteristics compared to a standard cell layout are the following:

a) Firstly, PLAs implement their logic function in 2-level form, which results in superior delay characteristics as long as the size of each PLA is bounded. On the other hand, in a standard cell implementation, considerable delay is incurred in traversing the different gates of the design.

b) In the PLA implementation scheme, local wiring is collapsed into a compact 2-level core, which is naturally crosstalk-immune. Hence local wiring delays are reduced.

c) Devices in the PLA core are minimum-sized, giving rise to extremely compact layouts. This is not the case for standard cell layouts.

d) In the PLA core, NMOS devices are used exclusively. As a result, devices can be placed extremely close together. However, in a standard cell layout, both PMOS and NMOS devices are present in each cell, and the PMOS-to-NMOS diffusion spacing requirement results in a loss of layout density.

e) Finally, the PLAs used are dynamic, making them faster than a static standard cell implementation.

The advantages can thus be seen to be in cross-talk immunity and favorable delay and area characteristics compared to traditional ASICs. By utilizing these novel PLAs interconnected in the manner of [2], all these benefits are exploited to implement fast, fully parallel LDPC code implementations. For each check node, $2d_c$ PLAs and $(log\ dc + 1)$ 2-input adders are used to implement its underlying operations. The checks and the variables are hard-wired with separate wiring in either direction. Also, uniform *5-bit* quantization is performed on the messages based on the channel information density. Accuracy can be improved by using non-uniform quantization that can be adaptively changed based on the evolution of the check and variable message densities.

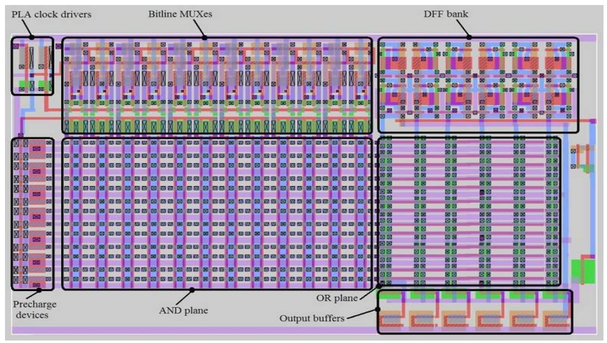


Figure 4: Structure of the PLA (layout) used in the check nodes

4. VLSI Implementation of LDPC CODECs

In order to utilize the IC area most efficiently, a decoder implementation with a square aspect ratio is sought. The proposed die floor plan is shown in Figure 5. The implementation consists of banks of check and variable (C/V) node clusters, arranged in a concentric configuration. White spaces in Figure 5 are reserved for clock drivers and control logic. There are four sets of banks shown in Figure 5, and denoted by S_1, S_2, S_3 and S_4, respectively. Each bank of C/V nodes consists of several C/V node clusters, shown in the right side of Figure 5. A cluster consists of a single check node, and several variable nodes. A typical high-rate code has a large number of variable nodes for each check node. For example, a rate 0.95 code has 20 variable nodes for each check node. Check node computations are more complex, as indicated by the larger area devoted to them in the figure. Denoting the number of clusters on one side of the lowest level by *a,* and the total number of check nodes by *m*, one can see that there are *r* rings in the concentric construction, where:

$$r = \left\lfloor \frac{\sqrt{(4+4a)^2 + 16(m-4a)} - (4+4a)}{8} \right\rfloor + 1 . \qquad (4)$$

The regularity inherent in the IC architecture of Figure 5 represents an input constraint for the code construction problem. In particular, the locality of a check node and several variable nodes in a cluster is exploited during the code construction process. Furthermore, in order to minimize the length of long wires between check and variable nodes, the codes are additionally constrained in such a way that nodes in the S_1 bank do not communicate (or communicate with a small number of edges) with nodes in the S_2 bank (likewise for the S_3 and S_4 bank nodes).

Prototype codes of this kind have been constructed, and custom IC implementations of these codes have been developed with excellent results as shown in Section 5,6.
The resulting design has the property that wiring is sparse and that long wires are minimized, by exploiting the regularity of the above IC architecture. At the same time, code performance is not significantly compromised by these constraints.

5. Example Related to the Concentric Construction

For the Concentric VLSI implementation described in the previous section, an LDPC code can be constructed based on the following set of requirements:

a) Variables and check nodes on opposite sides of the chip should not be mutually connected, or should have very few connections; this ensures that no wires cross the central region of the block.

b) Only nodes on the borderline of two neighboring sides are allowed to exchange messages during the decoding process; this ensures highly localized wiring.

Posed as constraints on the code design, these requirements take the following form. Assume that *V* is the set of variable nodes of the code, while *W* is the set of parity-check nodes. We seek a code with good error-correcting characteristics that allows for a partition of the set *V* into four subsets $V_1{}^c V_2{}^c V_3{}^c V_4$, approximately of the same size. If S_i denotes the subset of *W* that checks variable nodes in V_i, $i =$ 1,2,3,4 one should have:

$$|S_1 \cap S_2| < d_1, |S_3 \cap S_4| < d_1, |S_1 \cap S_3| < d_2,$$
$$|S_1 \cap S_4| < d_2, |S_2 \cap S_3| < d_2, |S_2 \cap S_4| < d_2$$

Here, $d_1{}^c\psi$and d_2 are non-negative integers such that $d_1 <<\psi$ d_2, with d_1 sufficiently small. In this setting, the vertices in $S_1{}^c\psi S_2{}^c\psi S_3{}^c\psi$ andψS_4 will be assigned to the four different sides of the chip, and there will be a very limited "exchange of information" between them. Furthermore, the variables in the intersection of sets S_1 and S_4, say, will be placed on the edge between the two corresponding sides.

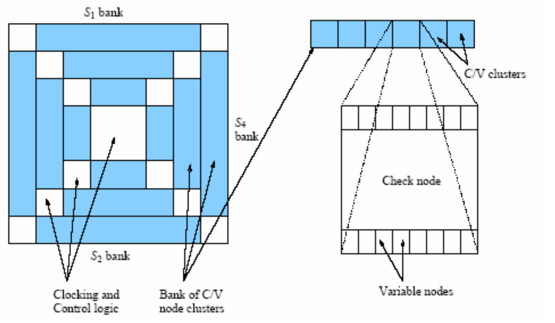


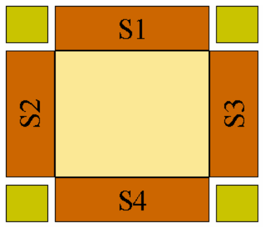
Figure 5: Concentric Implementation of LDPC codes

The placement within the S_i banks themselves can be governed by known proximity-preserving space-filling curves, such as the Hilbert-Paeno curve (for more details, the reader is referred to the full version of the paper).

We simulated the performance of a representative code of the class described above, with a parity-check matrix of the form H^* in Figure 6. Here, $d_1 = 0$, and the integers $i_1, i_2, ..., i_6$ are generated according to the construction methodology for large-girth array codes described in [6]. Figure 8 shows the BER curves for these lattice-based codes of rate ¾ for 16 iterations of message passing. The codes do not exhibit error floors up to BERs of the order of 10^{-8}, but show a deteriorated performance compared to the original array codes used for the construction. This is partly due to the increased value of the diameter of codes with parity-check matrices of the form H^*, compared to the codes from which they are derived from. More details about this phenomena can be found in the full version of the paper.

6. Estimation results

In order to estimate the improvements offered by the proposed decoder implementation, we used a 0.1 µ process. The delay and size estimates of the PLA were based on [2,3] and the size estimate of adders are as presented in [8]. An accurate delay/power description of both these hardware units based on SPICE simulations [7] was performed. It should be noted that in computing the size/delay/power estimates of adders and PLAs, wiring overhead, routing delay and the parity update operations at the checks are not accounted for. A minimal overhead is incurred on incorporation of these schemes. As an example, rate 3/4 codes with a block size of 8900, suited for a variety of applications are considered. The column weight d_v is 4 and the number of decoding iterations is 16. The tables show that the maximum achievable throughput is between *one and two orders* of magnitude higher than that demanded by most applications.



$$H^* = \begin{bmatrix} H_{1,1} & H_{1,2} & 0 & 0 \\ 0 & 0 & H_{2,1} & H_{2,2} \\ H_{3,1} & 0 & 0 & H_{3,2} \\ 0 & H_{4,1} & H_{4,2} & 0 \end{bmatrix}$$

Figure 6: Layout and parity-check matrix structure

$$H = \begin{bmatrix} P^{i_1} & P^{i_2} & P^{i_3} & P^{i_4} & P^{i_5} & P^{i_6} & 0 & 0 & 0 & 0 & 0 & 0 \\ P^{i_6} & P^{i_1} & P^{i_2} & P^{i_3} & P^{i_4} & P^{i_5} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P^{i_1} & P^{i_2} & P^{i_3} & P^{i_4} & P^{i_5} & P^{i_6} \\ 0 & 0 & 0 & 0 & 0 & 0 & P^{i_6} & P^{i_1} & P^{i_2} & P^{i_3} & P^{i_4} & P^{i_5} \\ 0 & 0 & 0 & P^{i_1} & P^{i_2} & P^{i_3} & P^{i_4} & P^{i_5} & P^{i_6} & 0 & 0 & 0 \\ 0 & 0 & 0 & P^{i_6} & P^{i_1} & P^{i_2} & P^{I_3} & P^{i_4} & P^{i_5} & 0 & 0 & 0 \\ P^{i_4} & P^{i_5} & P^{i_6} & 0 & 0 & 0 & 0 & 0 & 0 & P^{i_1} & P^{i_2} & P^{i_3} \\ P^{i_3} & P^{i_4} & P^{i_5} & 0 & 0 & 0 & 0 & 0 & 0 & P^{i_6} & P^{i_1} & P^{i_2} \end{bmatrix} \quad (9)$$

By lowering the clock speed, the power consumption can be brought down within acceptable limits for the required throughput. Hence, power dissipation does not represent a bottleneck for practical communication system applications. In order to compare the proposed approach with the standard-cell based implementation in [1], the estimates for a regular rate 1/2 code on 0.16 μ technology are provided as well. The parameters considered are n=1024, d_v=3, number of iterations equal to 64 and a power supply of 1.5 V. For a throughput of 1 Gbps, the side of the square chip based on the proposed implementation is 2.956 *mm* with a power dissipation of 0.723 Watts. This is a tremendous improvement on the area figures provided in [1] where a similar code dissipated 0.690 Watts with a chip size of 7.5*mm* x 7*mm*. An inclusion of all the overheads and also the timing recovery, the Serial-Parallel conversion and the Parallel-Serial conversion blocks is not expected to increase the side of the chip beyond 15% based on a very conservative estimate. It is observed that in the proposed implementation, the delay introduced by the variable nodes is almost three times smaller than that of a check node. It is therefore possible to further reduce the size of the chip by using multiplexers that would allow a single variable node unit to perform calculations for two variable nodes in a single clock cycle. This design strategy would include additional multiplexers, de-multiplexers and latches but reduce the number of variable node units by half.

7. Conclusions

A general high throughput VLSI architecture has been proposed that can be used to design LDPC decoder chips for specific applications as wireless communications, magnetic recording, optical communications etc. based on varied requirements. An important contribution is the clear-cut design approach which tightly couples the code design to the final VLSI layout. Besides, practical size and power constraints are met based on the preliminary estimates.

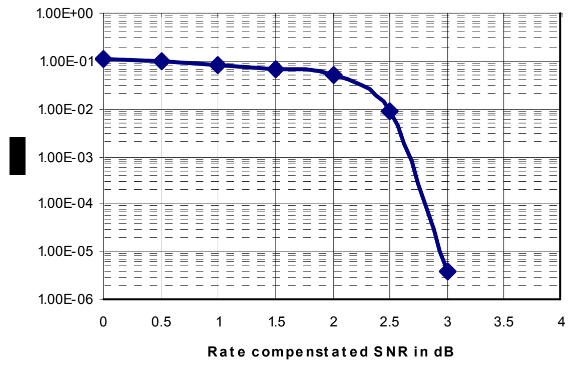


Figure 8: Error performance of regular rate-1/3 and rate ½ concentric codes

	Throughput (Gbps)	Side of chip (mm)	Power (Watts)
Flat out (max. duty cycle of 77.14%)	357.4476	6.198	30.3792
50% duty cycle (max. performance)	231.6876	6.198	21.1382
Lower clock for practical applications	2	6.198	4.2603

Table 4: Estimates for n=8900, Rate 3/4

References

[1] A.J.Blanksby and C.J.Howland, "A 690-mW 1024-b, Rate ½ Low-Density Parity-Check Code Decoder," *IEEE Journal of solid-state circuits, vol.37, No.3, Mar 2002*

[2] S. Khatri, R. Brayton, and A. Sangiovanni-Vincentelli, "Cross-talk immune VLSI design using a network of PLAs embedded in a regular layout fabric," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 412–418, Nov 2000.

[3] S. Khatri, R. Brayton, and A. Sangiovanni-Vincentelli, *Cross-talk Noise Immune VLSI design using regular layout fabrics*. Kluwer Academic Publishers, 2000. Research Monograph, ISBN # 0-7923-7407-X.

[4] F.R.Kschischang, B.J.Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no.2,pp. 498-519, Feb. 2001

[5] M. Mansour, and N. Shanbhag, "Memory-Efficient Turbo Decoder Architectures for LDPC Codes," *preprint*.

[6] O. Milenkovic, D. Leyba, D. Bennett, N. Kashyap, "New Partition-Regular Sequences and Array Codes of Large Girth," *accepted for presentation at the 44-th Allerton Conference on Control, Communications, and Computing, 2004.*.

[7] L. Nagel, "Spice: A computer program to simulate computer circuits," in *University of California, Berkeley UCB/ERL Memo M520*, May 1995.

[8] J.M.Rabaey, "Digital Integrated Circuits: A design Perspective," *Prentice Hall Electronics and VLSI series*

[9] T.Zhang and K.Parhi, "Join-(3,k)-Regular LDPC Code and Decoder/Encoder Design," *submitted to IEEE Trans. On Signal Processing*

[10] H. Zhong, and T. Zhang, "Design of VLSI Implementation-Oriented LDPC Codes," *preprint*.