Construction of Memory Circuits Using Unreliable Components Based on Low-Density Parity-Check Codes

Miloš Ivković
Department of Mathematics
University of Arizona
Tucson, AZ 85719, USA
Email: milos@math.arizona.edu

Shashi Kiran Chilappagari
Dept. of Electrical and Computer Eng.
University of Arizona
Tucson, AZ 85719, USA
Email: shashic@ece.arizona.edu

Bane Vasić
Dept. of ECE and Dept. of Mathematics
University of Arizona
Tucson, AZ 85719, USA
Email: vasic@ece.arizona.edu

***Abstract*— In this paper, we analyze storage circuits constructed from unreliable memory components. We propose a memory construction, using low-density parity-check codes, based on a construction originally made by Taylor. The storage circuit consists of unreliable memory cells along with a correcting circuit. The correcting circuit is also constructed from unreliable logic gates along with a small number of perfect gates. The modified construction enables the memory device to perform better than the original construction. We present numerical results supporting our claims.**

I. Introduction

Emerging nano-scale technologies will most likely lead to devices that will be less reliable than classical silicon based large scale integrated circuits. Such unreliability is inherent to technologies such as submicrometer designs, single-electron devices, and molecular electronics [1], [2], [3].

In this paper, we shall consider an unreliable component to be a component that is subject to *transient faults*, i.e., faults that manifest themselves at particular time steps but do not necessarily persist for later times [4]. In digital circuits, for example, transient faults due to noise (such as radiation or electromagnetic interference) are rapidly becoming a major cause for concern as the device size decreases.

There are two main directions of research in the area of systems made using faulty gates: computation and storage.

Study of computation by circuits with faulty gates is relatively old and rich. It started with von Neumann [5] in 1952 who showed that, under certain conditions, increased gate redundancy can lead to increased reliability of a circuit. The technique he introduced is called *multiplexing* and consists of two phases. In the first phase, the basic function of the circuit is performed in several copies of the circuit. In the second phase, referred to as the restoration phase, errors introduced in the first phase are corrected by random coupling of the copies, and the final outputs are considered to be "1" if majority of the considered copies are "1" and "0" otherwise.

A downside of this approach is that it leads to very high redundancy [5] (see also [6]). *Redundancy of the system* equals the ratio of the number of components within system to the minimal number of components (of the same type) needed to do computation performed by the system.

The multiplexing technique has been extensively studied. For example, Roy and Beiu [7] developed practical, small redundancy level schemes. Bhaduri and Shukla [8] studied multiplexing using the model of faulty gates based on statistical physics approach [9]. Nikolić *et al.* [10] proposed a version of multiplexing called Cascaded Triple Modular Redundancy (CTMR) where a sequence (cascade) of three input majority logic gates is used instead of one, very complex majority logic gate with high number of inputs.

Multiplexing can be seen as error control coding. For example, Triple Modular Redundancy (TMR) is nothing but a repetition code with code rate 1/3. Restoration organ of the multiplexing technique is in fact a error correcting decoder. Similar observation was made in [11]; for a tutorial on the subject see [12].

All the current fault tolerant construction are based on redundancy of the gates and it was shown that, in general, computation by faulty gates with non-zero computational capacity is not possible (see [13], [14], [15]). *Computational capacity* is the reciprocal of the minimum redundancy for which the probability of error can be made arbitrarily small [16]. The best known result for general model of computation is due to Speilman [17] who coupled ideas of von Neumann with Reed-Solomon codes.

The study of storage circuits made of unreliable components led to much more optimistic results. Taylor in [16] proposed construction of a storage circuit with non-zero computation capacity (or rather storage capacity) based on Low-Density Parity-Check (LDPC) codes (this construction was further studied by Kuznetsov [18]). Taylor also argues that no codes other than LDPC codes can achieve non-zero capacity. This is the reason we shall focus on LDPC codes.

Another positive result was obtained by Hadjicostis [4]. He was able to generalize Taylor's scheme to fault tolerant linear finite state machines.

A special case of the fact that no other codes but LDPC codes have non-zero computational capacity has been recently

rediscovered in [19] where Hamming codes were considered. In that paper it is shown that memory that uses TMR performs better than "Information Coding NanoBox" memory, which is essentially a Hamming code. The approach that we use (LDPC coding) is a generalization of TMR, and, considering results cited above and due to hardware complexity of other coding schemes, it is expected to perform better than any other coding scheme.

It is important to stress that non-zero capacity is possible only until the moment a codeword is transformed into some unencodable action, such as extracting message bits. For example, even if a systematic code is used in the construction of Taylor and Kuznetsov, at the end of the correcting procedure information will still be confined within a multiple of codeword copies with possibly some errors, i.e. discrepancy in content. So, the final step of extracting user information, must be done by a majority logic gate (or some equivalent circuit), which is by assumption faulty, resulting in probability of error grater than or equal to the probability of failure of the used majority logic gate. Therefore, it is reasonable to use reliable gates for the final step of decoding (see also [4] and [17]). Such gates can be realized, for example, by using larger transistor sizes.

In this paper, we propose a memory circuit scheme for LDPC decoder that is made mainly using faulty gates. The number of reliable gates needed is relatively small compared to the total number of gates. However, this memory circuit performs comparably to a memory circuit made entirely with reliable gates.

Analytical results on circuit reliability of Taylor [16] and Kuznetsov [18] are applicable in the case when codeword length is going to infinity. This was needed to assume that the size of the minimal cycle in the associated Tanner graph goes to infinity, yielding a possibility to analytically estimate codeword error probability. Absence of a theory for the finite length message-passing iterative decoding of LDPC codes, even in the case of perfect gates, forces us to run extensive numerical simulations to support our claims.

The rest of the paper is organized as follows. In Section II we present our proposed memory circuit construction. Supporting numerical results are given in Section III. We conclude in Section IV.

II. Memory circuit construction

Consider a regular binary LDPC code of length n. Let the word to be stored consist of bits $(x_1, x_2, \ldots, x_n)$ referred also as variables. Parity check matrix H with dimension $(n-k)\times n$ ($k < n$), of a regular LDPC code has K 1's in each row and J 1's in each column. Rows correspond to variables, and columns correspond to checks, i.e, each variable bit x_i, $1 \leq i \leq n$ is involved in J parity-check equations by:

$$[x_1, x_2, \ldots, x_n]H^T = [c_1, c_2, \ldots, c_{n-k}]$$

where all operations are in binary field and c_t corresponds to the value of t-th parity-check sum for $1 \leq t \leq n-k$. Vector $[c_1, c_2, \ldots, c_{n-k}]$ is called *syndrome*. Parity check c_t is said to be *satisfied* if $c_t = 0$ and *unsatisfied* if $c_t = 1$.

Taylor's construction is a decoding scheme for LDPC codes. In this scheme [16] (see also [18], [4]), each bit x_i is replaced with J bit-copies $\{x_i^1, x_i^2, \ldots x_i^J\}$ stored in J registers (all bit-copies initially have the same value). Registers are made from memory cells that are considered to be unreliable. New estimates of each of these copies is obtained by using one combination of $J-1$ checks. Note that there are exactly $\binom{J}{J-1} = J$ combinations. The estimates are obtained as follows.

1) Evaluate parity checks for each bit-copy (exclude one different parity check from the original set of checks for each bit-copy).
2) Flip the value of a particular bit-copy if half or more of the parity checks are unsatisfied.
3) Iterate (1) and (2).

A block diagram of this iterative decoding scheme is given in Fig. 1. There are a total of $J-1$ parity checks for each bit copy. The decision element in this case is a majority logic gate whose output is 1 if half or more of the parity checks are nonzero. The correction is accomplished by XOR gate at the bottom that has as inputs the output of the majority logic gate and the previous value of the bit-copy.

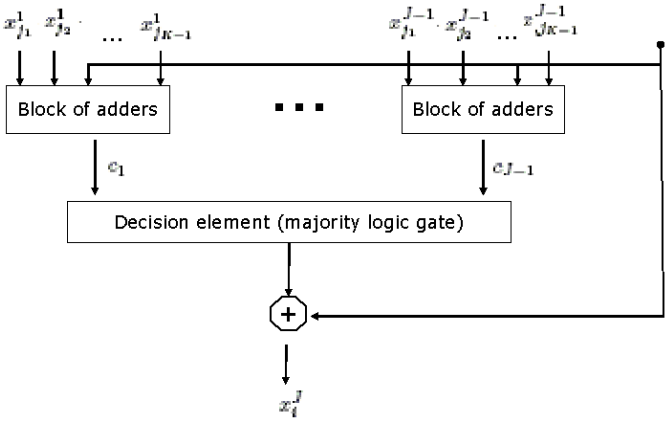

Fig. 1. Original Taylor's scheme

Note that in this iterative decoding scheme, each parity check requires $K-1$ input bits (other than the bit-copy we are trying to estimate). Since each of these input bits has J different copies, there are J copies for each particular bit that can be used for estimating x_i^k, k-th copy of bit x_i. When estimating a copy x_i^k of bit x_i using an estimate of bit x_j, we use the bit-copy x_j^t in whose evaluation the parity check involving x_i is omitted. Note that the whole scheme corresponds to a version of, so called, GallagerB decoding algorithm on Binary Symmetric Channel [20], [21].

In the rest of this section we present modifications to this construction that we are proposing.

If we exclude a bit copy that we are trying to estimate from the parity check, this parity check is not calculating whether that bit copy is correct, but a value of the bit copy itself. So each of $J-1$ checks involved in correcting one bit needs

to have one input less. In this case the voter will output the new estimate for this bit copy, so the bottom XOR gate is not needed neither (see Fig. 2). This not only makes construction more efficient, but also more dependable, since each of this XOR gates is considered to be faulty.

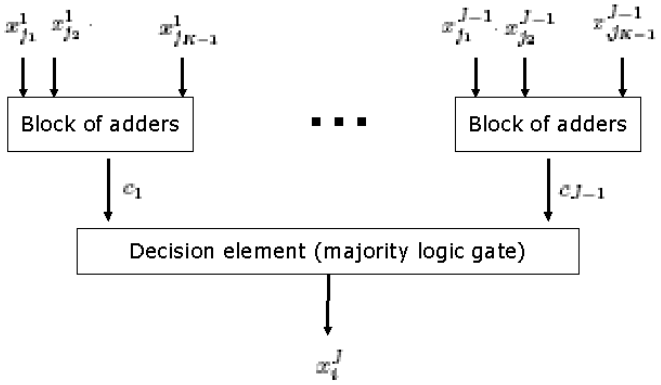

Fig. 2. The proposed first modification

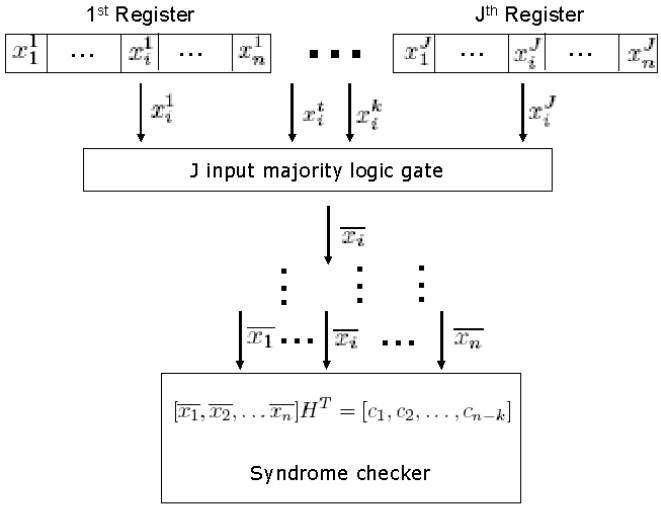

Fig. 3. The syndrome checker

In the case when $J-1$ is an even number, tie can be broken either arbitrarily (as suggested by Taylor) or by the received value of the bit that is being corrected, as it is usually done in message-passing algorithms [21]. In the case when J is small ($J = 3, 4, 5 \ldots$) ties are relatively frequent and breaking them arbitrarily can hurt decoder's performance, so this is an important issue. Note also that in the case when memory cells are unreliable it does not make sense to use the originally stored value. We opted to break ties by using the previous value of the bit being corrected.

The second important modification to the Taylor's construction that we propose is introducing a syndrome checker after every iteration. If the valid codeword is reached, decoded codeword is stored in all the registers. So the modified decoding procedure is as follows:

1) Evaluate $J-1$ estimates for each bit-copy (exclude one different parity check from the original set of checks for each bit-copy).
2) Update the value of bit copy based on majority of J-1 estimates (and the previous value of bit, if needed).
3) Calculate estimates $\overline{x}_i$ as outputs of majority logic gates with inputs $x_i^1, x_i^2, \ldots x_i^J$. See Fig. 3.
4) If the syndrome for $\overline{x}_1, \overline{x}_1 \ldots \overline{x}_n$ is a zero vector $\overline{x}_1, \overline{x}_1 \ldots \overline{x}_n$ is stored in all the registered, otherwise iterate (1), (2) and (3). Syndrome calculation is done by *perfect* gates.

As already mentioned in the step 4. above, we shall assume that syndrome checker is perfect i.e. that it is not made of faulty gates. In this way we are adding n perfect majority logic gates, $n-k$ perfect K-input XOR gates and one $n-k$-input OR gate needed for calculating whether all syndrome values are zero. So the total number of faulty gates used is $nJ(J-1)$ XOR gates and nJ majority logic gates. Compared to the total number of gates used, number of perfect gates is not big. It should be noted that the original Taylor's construction also requires perfect majority logic gates at the moment when information needs to be retrieved from the registers as mentioned in the introduction. Thus incorporating a certain number of perfect gates is always required.

III. Numerical Results and Discussion

In this section, we assume that the whole construction is made of NAND gates. The use of only NAND gates in extremely large-scale integration (XLSI) architectures may be justified by the fact that construction is simplified through the use of identical repeating sub-units [10]. Fig. 4 and Fig. 5 show one possible realization of a three-input majority logic gate and five-input XOR gate respectively. These are the realizations we have used in our simulations.

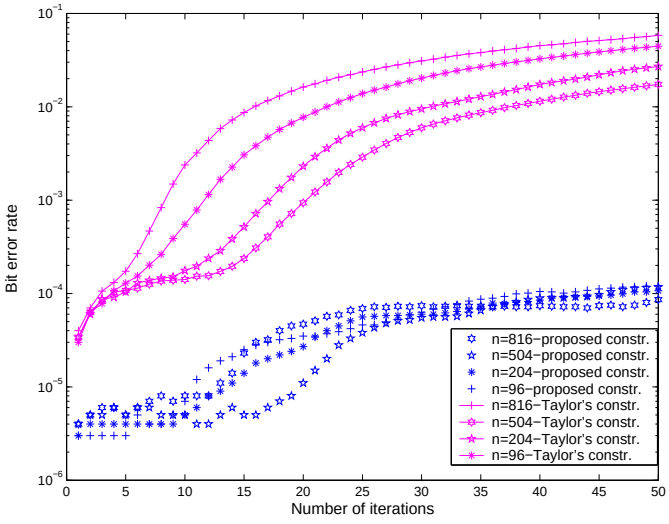

Fig. 6. Comparison of Taylor's construction and proposed construction

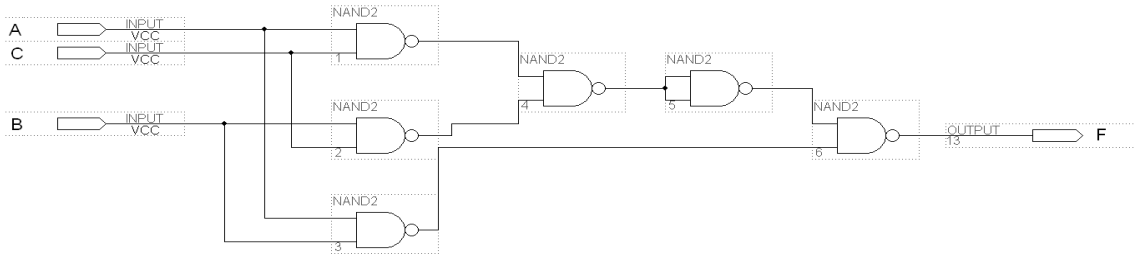

Fig. 4. Three input majority logic gate

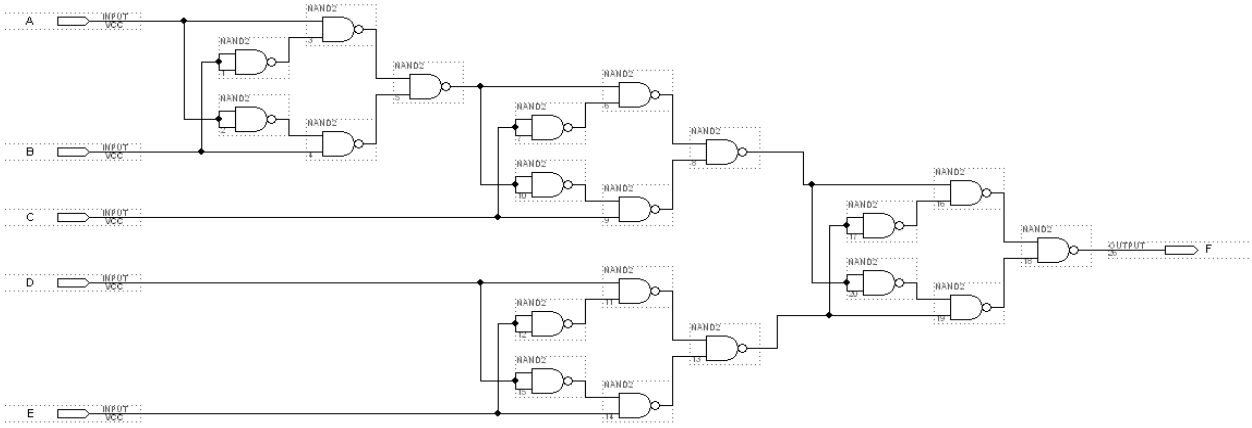

Fig. 5. Five input XOR gate

We also assume that gates fail independently of each other, that defects are not permanent, i.e. gate that malfunctioned in one iteration may give correct output in subsequent iterations and that failure occurs by flipping the correct result with some probability, i.e. if the correct result is 1 but the gate gives 0 and vice versa. We denote this probability of failure of a NAND gate by ϵ_{NAND}. We assume that connections between gates are perfect. The errors in connections can be equivalently represented as errors of the proceeding gate, (e.g. [13]) , in a way very similar to how encoder errors are represented as errors of the channel. The memory components are assumed to fail independently with probability ϵ_{mem}.

To illustrate our findings, we consider LDPC codes constructed by MacKay [23]. We present results for four column-weight-three codes of same rate, 0.5, but different lengths: 96, 204, 504 and 816.

Fig. 6 compares the proposed construction with Taylor's construction for different codes. In this case $\epsilon_{\mathrm{mem}} = 5 \times 10^{-3}$ and $\epsilon_{\mathrm{NAND}} = 5 \times 10^{-5}$. From the figure it is clear that the proposed construction outperforms Taylor's construction and that the gain is considerable even at $\epsilon_{\mathrm{mem}} = 5 \times 10^{-3}$.

Fig. 7 illustrates the effect of ϵ_{NAND} on the bit error rate (BER) performance of the memory circuit for a fixed ϵ_{mem}. The code considered is of length 204 and $\epsilon_{\mathrm{mem}} = 0.4\%$. Other codes with different values of ϵ_{mem} show similar behavior. It can be seen that the performance with faulty gates is of the same order as that with perfect gates.

Fig. 8 illustrates the effect of ϵ_{mem} on the performance of the proposed memory circuit. The code considered is of length 96 and the correcting circuit is considered entirely perfect.

Notice that obtaining theoretical bounds on error probability is elusive for the time being. This is because the general theory of LDPC decoding is still not developed in full. For the theoretical prediction of a faulty decoder performance we must know the performance of a reliable decoder (for a given code). This requires classification of trapping sets. This is a field of great current interest [24], but it is outside of the scope of this paper.

IV. Conclusion

We showed that memories made of faulty gates and a relatively small number of perfect gates can perform very well. The construction is practical both in terms of redundancy and performance.

We supported our claims by extensive numerical simulations. Note that proposed construction does not depend on a

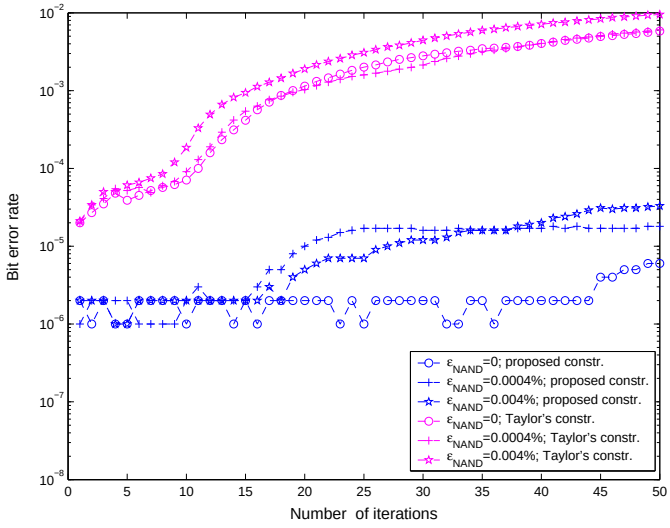

Fig. 7. Performance comparison for different values of ϵ_{NAND}

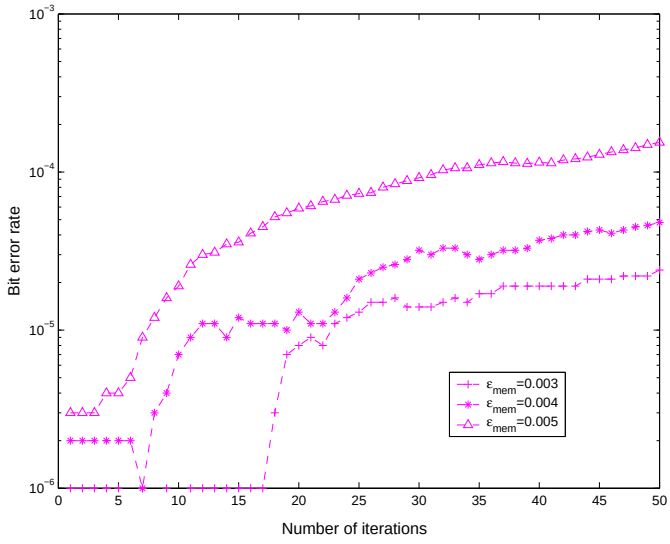

Fig. 8. Performance comparison for different values of ϵ_{mem}

specific gate realization. Future work includes investigation of analytical bounds on performance.

ACKNOWLEDGMENT

The authors would like to thank P. Hio for providing realizations of majority logic gates and XOR gates from NAND gates. This work was supported by the NSF under Grant CCR-0208597.

REFERENCES

[1] P. Larsson and C. Svensson, "Noise in digital dynamic CMOS circuits," *IEEE J. Solid-State Circuits* , vol. 29, no. 6, pp. 655662, Jun. 1994.

[2] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," *Proc. Int. Conf. Computer-Aided Design (ICCAD 1996)*, pp. 524531, 1996.

[3] I. Amlani, A. O. Orlov, G. Toth, G. H. Bernstein, C. S. Lent, and G. L. Snider, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284, pp. 289 - 291, Apr. 1999.

[4] C. N. Hadjicostis and G. C. Verghese, "Coding Approaches to Fault Tolerance in Linear Dynamic Systems," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 210-228, Jan. 2005.

[5] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, C.E. Shannon and J. McCarty, eds., Princeton University Press, pp. 43-98, 1956.

[6] A. Sadek, K. Nikolic and M. Forshaw, "Parallel information and computation with restitution for noise-tolerant nanoscale logic networks," *Nanotechnology* vol. 15, pp. 192-210, 2004.

[7] S. Roy and V. Beiu, "Majority Multiplexing: Economical Redundant Fault-Tolerant Designs for Nano Architectures," preprint, available at: http://www.eecs.wsu.edu/ vbeiu/Publications/2005%20T_NANO.pdf

[8] D. Bhaduri and S. K. Shukla, "NANOLAB: A Tool for Evaluating Reliability of Defect-Tolerant Nano Architectures," preprint, available at: http://fermat.ece.vt.edu/Publications/online-papers/Nano/nanolab_journal.pdf

[9] R. Bahar, J. Mundy, and J. Chen, "A probability-based design methodology for nanoscale Computation," *Proc. Intl. Conf. Computer Aided Design ICCAD'03*, San Jose, CA, USA, pp. 480-486, Nov. 9-13, 2003.

[10] K. Nikolic, A. Sadek, and M. Forshaw, "Architectures for reliable computting with unreliable nanodevices," it Proc. IEEE-NANO, pp. 254-259, 2001.

[11] M. Sipser and D. Spielman, "Expander Codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710-1723, Nov. 1996.

[12] P.Gács, "Lectures on Reliable Cellular Automata," http://citeseer.ist.psu.edu/20653.html

[13] R. L. Dobrushin and S. I. Ortyukov, "Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.*, 13:59–65, Jul. 1977.

[14] P. Gács and A. Gál, "Lower bounds for the complexity of reliable Boolean circuits with noisy gates," *IEEE Trans. Information Theory*, vol. 40, pp. 579-583, Mar. 1994.

[15] N. Pippenger, G. D. Stamoulis and J. N. Tsitsiklis, "On a lower bound for the redundancy of reliable networks with noisy gates," *IEEE Trans. Inf. Theory* vol. 37, pp. 639-643, May 1991.

[16] M. Taylor, "Reliable Information Storage in Memories Designed from Unreliable Components," *Bell System Technical Journal* vol. 47, pp 2299-2337, Dec. 1968.

[17] D. Spielman, "Highly Fault-Tolerant Parallel Computation," *IEEE Conference on Foundations of Computer Science*, pp 154-163, 1996.

[18] A. Kuznetsov, "Information Storage in a Memory Assembled from Unreliable Components," *Problems of Information Transmission* vol. 9, pp. 254-264, Jul. 1973.

[19] A. KleinOsowski, P. Ranganath, M. Subramony, V. Rangarajan, K. KleinOsowski, D. Lilja,: "The NanoBox: A Self-Correcting Logic Block for Emerging Process Technologies with High Defect Rates," Laboratory for Advanced Research in Computing Technology and Compilers Technical Report No. ARCTiC 03-02, June, 2003.

[20] R. Gallager, *Low-Density Parity-Check Codes* Cambridge, Massachusttes, MIT Press, 1963.

[21] R. Urbanke, *Iterative Coding Systems*, unpublished notes. http://lthcwww.epfl.ch/publications/index.php

[22] http://www-cad.eecs.berkeley.edu/software.html

[23] http://www.inference.phy.cam.ac.uk/mackay/codes/

[24] T. Richardson, "Error floor for LDPC codes," *in Proc. of 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003.