A Space-Time Redundancy Technique for Embedded Stochastic Error Correction

Chris Winstead, Yangyang Tang, Emmanuel Boutillon, Christophe Jego, Michel Jezequel

A Space-Time Redundancy Technique for Embedded Stochastic Error Correction

Chris Winstead*, Yangyang Tang†, Emmanuel Boutillon†, Christophe Jego‡ and Michel Jezequel§
*Dept. of Electrical and Computer Engineering, Utah State University, Logan, Utah 84322
Email: winstead@engineering.usu.edu
†Université de Bretagne Sud, UMR CNRS 6258 Lab-STICC, Lorient, France
Email: yangyang.tang, emmanuel.boutillon@univ-ubs.fr
‡Institut Polytechnique Bordeaux, UMR CNRS 5218 Lab-IMS, Bordeaux, France
Email: christophe.jego@enseirb-matmeca.fr
§Institut TELECOM/TELECOM Bretagne, UMR CNRS 6258 Lab-STICC, Brest, France
Email: michel.jezequel@telecom-bretagne.eu

***Abstract*—An error-correction algorithm, referred as to Low Density Parity Check (LDPC) stochastic decoding technique, has recently been introduced for implementing iterative LDPC decoders in logic technologies with a high rate of transient faults. In this work, a modified algorithm that includes a feedback mechanism is first presented. A temporal majority logic is also applied at the decoder's output, providing an additional dimension of redundancy. By comparison to Gallager-A decoding method, the combination of feedback with temporal redundancy is shown to significantly increase the decoder's resilience against a high rate of internal upsets as a gain of up to three orders of magnitude.**

I. Introduction

As digital technologies approach the limits of planar integration, device-level reliability has emerged as a critical concern. Densely integrated CMOS circuits are increasingly affected by thermal upsets which induce momentary, transient signal faults in digital systems. The problem of transient upsets is compounded with the increased use of three-dimensional integrated circuits, and with the emergence of "post-CMOS" nano-scale devices that perform computations using a very small number of electrons, or even a single electron. Because of these concerns, it has been proposed to embed error-correcting logic within integrated digital systems.

Embedded error correction may be used to mask permanent circuit defects as well as transient faults. Unfortunately an embedded error correcting decoder has to be designed using the same error-prone devices as the functions to correct. This introduces the problem of synthesizing decoders that are able to detect and/or correct faults in their own operation. There is a long history of research on this problem, with solutions in the form of "self-checking checkers." Self-checking logic circuits are traditionally designed using formal logic methods, where it can be proved that the circuit detects or corrects up to a fixed number of faults. Some researchers also studied the resilience of iterative decoding algorithms implemented with faulty internal logic [1]. In this paper, we propose an alternative analysis style based on *probability signal flow*.

Probability Signal Flow (PSF) analysis was previously proposed as a method to embed Low-Density Parity-Check (LDPC) codes [2] for fault compensation in nano-scale digital logic [3]. The PSF approach was also applied at the circuit level, resulting in a robust error correction method known as Restorative Feedback (RFB) [4]. The RFB method is logically identical to the traditional method of Triple Modular Redundancy (TMR) [5]. However it was shown via PSF analysis that the RFB method achieves a much lower error rate than TMR in the presence of internal transient upsets. On the contrary, an LDPC Stochastic Decoding (LSD) technique has been developed to cope with internal high-rate fault by requiring redundancy of one under certain constraints [6].

In conventional LDPC code applications, the decoder is designed to operate on very large codewords, usually thousands of bits. Shorter word sizes are more appropriate for digital logic systems such as microprocessors and arithmetic circuits. In this paper, we extend the LSD approach by applying a space-time technique to refine its error-correction capacity of embedded decoders for short codewords. We consider the use of temporal majority logic at the decoder's output. This introduces time-redundancy that is shown to be effective at rejecting transient errors that originate within the decoder's logic.

The remainder of this paper is organized as follows: Section 2 reviews the decoding methods and then present the modified LSD method and the space-time technique as well. Experimental results are given in Section 3. Discussion and conclusions are given in Section 4 and Section 5, respectively.

II. Decoding Methods and Space-Time Technique

In this section, we first review a hard-decision bit-flipping algorithm, Gallager-A method, and the LSD method as well. Then, the modified C-element circuit and the modified LSD algorithm are presented. The space-time technique is detailed afterwards.

A. LDPC decoding techniques

LDPC decoding methods are traditionally described as message-passing on the Tanner graph [7] of a sparse parity-check matrix associated to a LDPC code. As usual, the

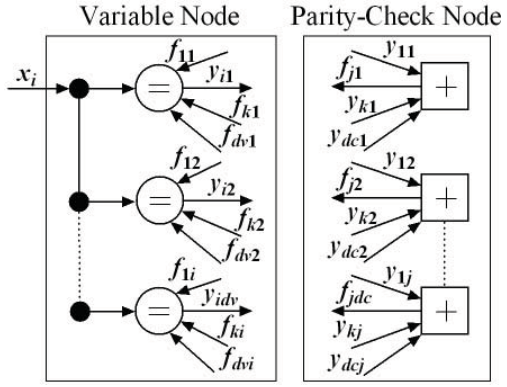

Fig. 1: Structure of the *variable* and *parity-check* nodes that comprise an LDPC decoder. The extrinsic information principle is applied, so that an output on a given edge j is computed from all inputs *except* j.

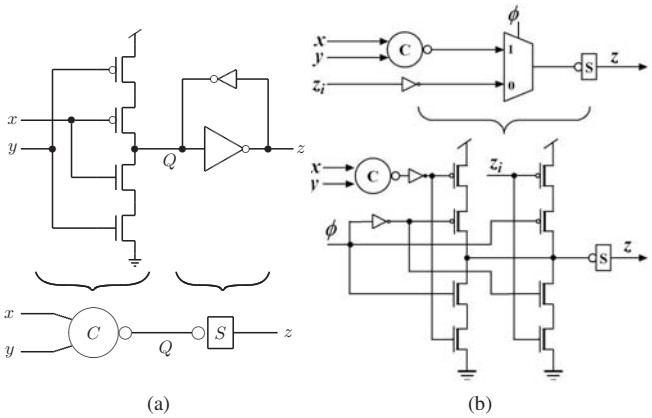

Fig. 2: (a) A standard Muller C-element circuit, divided into C-not and S components. The C-not gate detects whether the inputs are equal, and the S gate acts as an inverting state memory. (b) A modified Muller C-element with two-phase operation. Phase $\phi = 0$ allows initialization of the S latch. Phase $\phi = 1$ enables normal operation of the C-not gate.

bipartite graph contains two sets of nodes – variable nodes (or symbol nodes) v_i in degree d_v and parity-check nodes p_j in degree d_c. Without loss of generality, we limit our discussion to the class of regular LDPC codes, in which all variable nodes have equal degree d_v and all check nodes have equal degree d_c. During the decoding process, binary messages are exchanged between the two sets of nodes. To simplify notation in this paper, we index incoming and outgoing messages locally at a particular node. For instance, when referring to a variable node v_i, the sets of incoming and outgoing messages are written as f_{ki} and y_{ik}, respectively, for local edge indices $k \in \mathcal{V} = \{1, \ldots, d_v\}$. The channel-side input is written $f_{0i} = x_i$. Similarly when referring to a check node p_j, the incoming and outgoing messages are written y_{kj} and f_{jk}, respectively, for local edge indices $k \in \mathcal{P} = \{1, \ldots, d_c\}$. This notation is indicated in Fig. 1. When there is no ambiguity, the i and j subscripts will be omitted, e.g. f_k, y_k, etc.

For all decoding methods discussed in this paper, the parity-check nodes perform a modulo-2 summation over their input messages, as is usual with stochastic and Gallager-style decoders [8]–[10]. Specifically, the outgoing message for each $k \in \{0, \ldots, d_c - 1\}$ is $f_{jk} = \oplus_{m \in \mathcal{P}_k} y_{mj}$ where $\mathcal{P}_k$ is the $\mathcal{P}$ excludes k. These operations are implemented by using a cascade of XOR gates. In the remainder of this paper, we discuss three decoding algorithms that use this parity-check rule, but differ in the processing performed at the variable nodes.

B. The Gallager-A method

Gallager introduced a collection of bit-flipping algorithms for LDPC decoding in binary channels [2]. In the case of embedded decoders for digital logic computation, we assume a binary symmetric channel (BSC) model. For this channel, the most appropriate of Gallager's original algorithms is the Gallager-A method, described as follows. For each variable node, each outgoing message is initially equal to the received bit x_i. In subsequent iterations, the message on edge k is

$$y_k = \begin{cases} 1, & \prod_{m \in \mathcal{V} \setminus k} f_{mi} = 1 \\ 0, & \prod_{m \in \mathcal{V} \setminus k} \overline{f_{mi}} = 1 \\ x_i, & \text{otherwise.} \end{cases}$$

C. The modified C-element circuit

Similar to the RFB method described previously, the LSD method uses Muller C-elements [11], which are increasingly used for fault-masking in digital circuits [3], [4], [9], [10], [12], [13]. A standard binary C-element circuit is shown in Fig. 2a. In this circuit, the "C-not" gate detects whether the inputs are equal, with an output given by

$$Q = \begin{cases} \overline{x}, & \text{if } x == y \\ Z, & \text{otherwise,} \end{cases} \quad (1)$$

where Z denotes a high-impedance output state. Whenever $Q \neq Z$, the C-not gate actively drives the S latch, overpowering its state. When $Q = Z$, the state of S is maintained via weak feedback. To implement C-element based decoding algorithms, it is necessary to use the modified C-element circuit shown in Fig. 2b. This circuit was introduced in [4] to support error-correcting operations in the RFB method. The modified C-element works in two phases, called "initialization" and "restoration." During the initialization phase, the C-not gate is disconnected so that the S latch can be set to a known initial state. During the subsequent restoration phase, the C-element is activated. The two-phase behavior can be applied to implement Gallager-style error-correction, as shown in the following subsections. The benefit of this approach is that C-elements are able to reject transient upsets that occur internally within the decoder. We suppose the inputs x and y are copies of a single logic value. Then if the C-element is initialized to a correct state, any subsequent transient upsets in x or y are rejected.

D. C-element methods

In this subsection, the LSD [6] and the LSD with feedback (LSDfb) methods that employ C-element logic to suppress internal transient faults are presented. The LSD is characterized by the processing steps applied at the variable nodes. At each variable node v_i, we associate a set of C-element gates C_k, $0 \leq k < (d_v - 1)$. Each C-element gate C_k contains a single-bit storage element c_k. The error correction algorithm is:

1) Initialize $y_k = x_i$, for all $k \in \mathcal{V}$.
2) Compute $f_{ji} = \oplus_{m \in P_{j \setminus i}} y_{mj}$ for all $j \in \mathcal{P}$.
3) Initialize each C-element memory as $c_k = f_m$, where $m = (k + d_v - 1) \bmod d_v$.
4) The C-element's port connections are as follows. For C_0, the inputs are f_0 and f_1, and the output is c_0. For the remaining C_k, the inputs are c_{k-1} and f_{k+1}, and the output is c_k.
5) Iterate steps 2 and 4 during a fixed number of iterations. The initialization in step 3 is performed only during the first iteration.
6) The corrected output is $z_i = c_{(d_v-1)}$.

The LSD algorithm may be modified by adding a restorative feedback connection, which has been previously shown to suppress internal transient faults [4]. This modification is applied to Step 4, which is revised to read

4. The C-element's port connections are as follows. For C_0, the inputs are y_k and f_1, and the output is c_0. For the remaining C_k, the inputs are c_{k-1} and f_{k+1}, and the output is c_k.

This modification will be improved the decoder's resilience to internal faults when time-redundancy is applied to the output bits, z_i.

The LSD and LSDfb methods can be regarded as circuit-level designs for variable nodes in a LDPC decoder. Cascaded C-element sets are employed in both methods. In Fig. 3 and Fig. 4, the architectures of the variable node for LSD and LSDfb are detailed, respectively. During the decoding process, the modified C-element with phase operation is used to implement the algorithm's initialization phase. The feedback modification for the LSDfb method is also given in Fig. 4.

E. The space-time technique

In order to further improve the decoder's resilience to internal faults, a temporal majority mechanism is applied at the decoder's output. The temporal-redundancy strategy is expected to improve fault suppression because the LSD and LSDfb methods are derived from stochastic decoders, in which messages are considered as stochastic *streams* [9], [10]. In a stochastic decoder, the output decisions are rendered by performing a time-majority on the output streams. Since the LSD and LSDfb decoders are already performing iterations, the time-voting can occur during the last few iterations.

To validate the effect of temporal redundancy, the decoder's output from $c_{(d_v-1)}$ are processed into a majority operation, $\text{Maj}(c_{(d_v-1)}, l)$ where l denotes the number of samples. In our work, we consider a three-of-five voter as shown in Fig. 5. This structure is convenient for logic-level simulation, but other strategies are also possible, as discussed in Sec. IV.

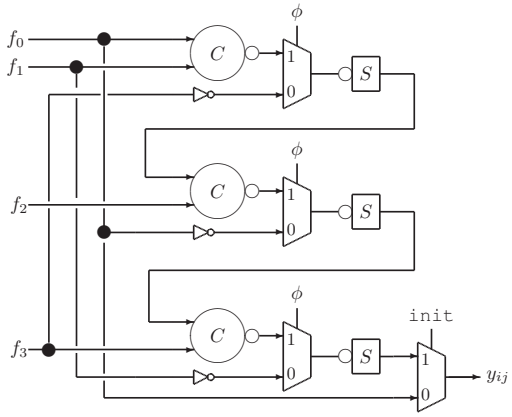

Fig. 3: LSD circuit, $d_v = 4$.

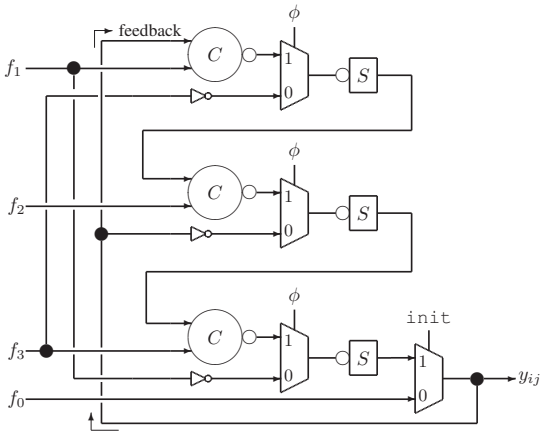

Fig. 4: LSDfb circuit, $d_v = 4$.

III. Simulation Results

In this section, we apply the embedded decoders to achieve fault-resilience in a digital logic circuit proposed in [6], as shown in Fig. 6. A logic function $F(x)$ is implemented using a digital technology that is subject to errors at its output. The original function F is complicated by the addition of a redundant parity-generator module, $E \cdot F$, where E represents the encoding function that generates parity bits codeword space at the output of F as in [3]. The *systematic* output word s with a length K from F is then concatenated with the

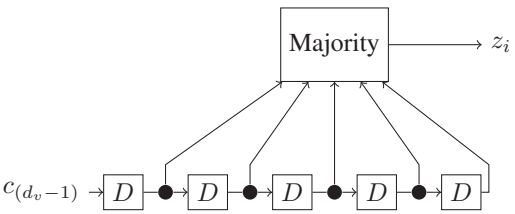

Fig. 5: Structure for a temporal three-of-five majority voter.

parity outputs r from $E \cdot F$, yielding a complete codeword $[s\ r]$ of length N. According to the code's H matrix, the embedded decoder is synthesized by instantiating arrays of variable and check nodes circuits, which are indicated by $\boxplus$ and $\circleddash$, respectively. The system's final decisions z are taken from the variable node circuits, otherwise from the additional time voter if the space-time redundancy approach is applied.

The Gallager-A, LSD and LSDfb methods were simulated using a systematic regular (3, 6) LDPC code. To conform with typical word sizes used in electronic devices, the codeword size is $N = 64$, corresponding to 32-bit data words. In our simulations, the output bits s and r from $F(x)$ and $E \cdot F(x)$, respectively, are assumed to have a uniform independent error probability of ϵ. The logic gates that comprise the decoding circuit are assumed to be faulty boolean operations, which have a uniform error probability of α. In our simulations, the temporal-majority component is assumed to be a reliable output interface circuit.

The Bit Error Rate (BER) performance results with no time-redundancy are shown in Fig. 8. With no time-redundancy, all methods achieve poor performance. In the high-α case shown in Fig 8(b), the Gallager-A method is seen to degrade completely with increased iterations. The results using time-redundancy are shown in Fig. 9. Unlike the degradation with increased iterations in Gallager-A, the LSD methods introduce a significant improvement in terms of BER performance for the high-α case. Time-redundancy is not effective when applied to the traditional Gallager-A decoder, because its degradation is too severe. In addition, the feedback approach is shown to be beneficial in the high-α case as well.

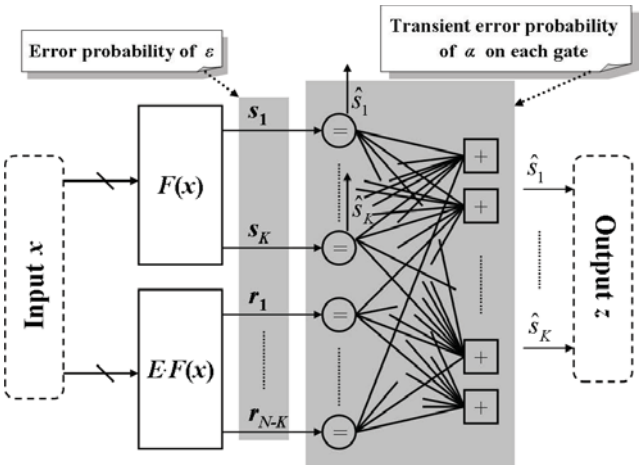

Fig. 6: The proposed fault-tolerant architecture, the shade area representing the proposed embedded decoder.

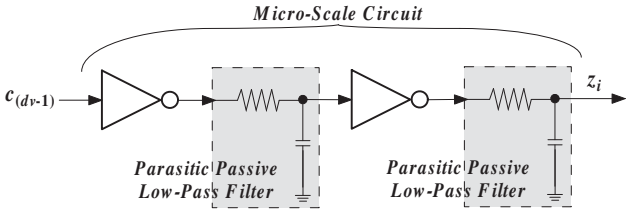

Fig. 7: Passive time-redundancy can be obtained by a low-pass filter. This approach may be implemented, for instance, with the native R-C parasitics in a large-size output buffer.

IV. Discussion

A. Redundancy hardware cost of the proposed architecture

To evaluate the hardware redundancy cost of the proposed architecture, we suppose that $F(x)$ is a large block of crossbar logic (i.e. gate array logic) representing a minimized sum-of-products expression [14]. The crossbar fabric may consist, for example, of a layer of AND operations followed by a layer of OR operations, which may be used for flat truth-table synthesis. Although this style of logic is generally not optimal in terms of device count, it provides a precise strategy for synthesizing the auxiliary function $E \cdot F$. For arbitrary functions F, the average total logic gate-complexity needed to synthesize $E \cdot F$ is close to the average number of logic gates needed to synthesize F. Hence the proposed method requires an average logic gate redundancy of approximately two, plus the additional logic gates.

B. A feasible time-redundancy implementation

In order to demonstrate the benefits of time-redundancy in the LSD kind decoders, it is necessary to assume that the time-voting circuit can be implemented with error-free components. This assumption merits further examination. In the study of "post-CMOS" nano-scale electronics, it is common to imagine a hybrid system in which logic operations are built from faulty nano-scale gates, while output interfaces are implemented using larger, more reliable CMOS devices. For any fault-masking method, it is critical to assume that the final read-out circuits are more reliable than the other internal components, or else the system's overall reliability will be limited by the interfaces themselves. It is furthermore possible to imagine alternative circuit solutions for implementing time-redundancy. For example, if the logic technology operates at a much higher speed than the output interface devices, then the time-redundancy may be implemented via passive RC filtering in the output interface. An example of this is shown in Fig. 7, where low-pass filtering is performed by parasitic elements within two inverter buffers. This arrangement achieves an average-and-threshold operation, which is functionally equivalent to the time-voting mechanism used in our simulations.

V. Conclusion

The main contributions of this work are the introduction of a space-time redundancy technique for embedded error-correction, and its implementation at the circuit level. When temporal redundancy is applied at the decoder's output, the new C-element based techniques were shown to be resilient to errors that appear within the decoder's logic, whereas the traditional Gallager-A algorithm proved less efficient under these conditions.

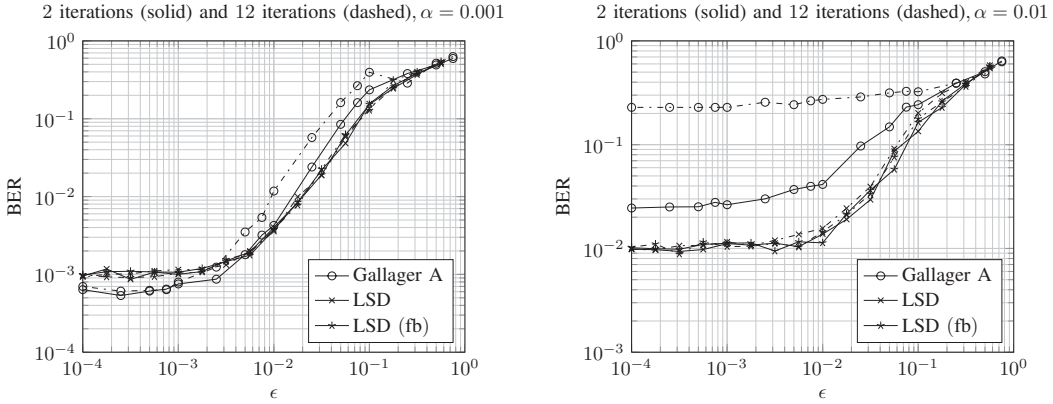

Fig. 8: BER results in function of iteration number without applying time-redundancy. When the intrinsic gate-error rate (α) is high, the Gallager-A performance worsens with increased iterations. The LSD methods do not exhibit this degradation.

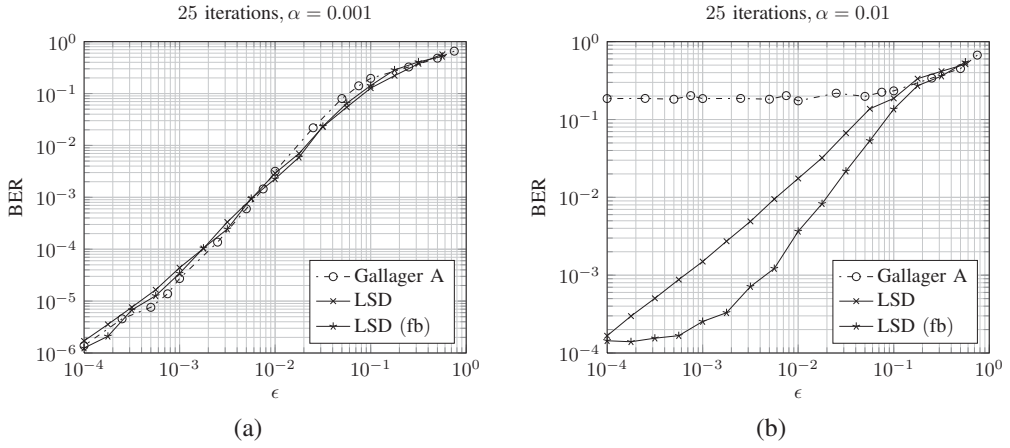

Fig. 9: BER performance results with time-redundancy applied at the decoder's output. The LSD methods achieve improved performance when α is high. The feedback approach is also shown to be beneficial in the high-α case.

ACKNOWLEDGMENT

This work was supported by the US National Science Foundation under award ECCS-0954747.

REFERENCES

[1] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 11, pp. 2438 –2446, nov. 2007.

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

[3] C. Winstead and S. Howard, "Probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *IEEE Transactions on Circuits and Systems II – Express Briefs*, vol. 56, no. 6, pp. 484–488, June 2009.

[4] Chris Winstead, Yi Luo, Eduardo Monzon, and Abiezer Tejeda, "An error correction method for binary and multiple-valued logic," *Multiple-Valued Logic, IEEE International Symposium on*, pp. 105–110, 2011.

[5] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J.*, vol. 6, pp. 200–209, 1962.

[6] Yangyang Tang, Chris Winstead, Emmanuel Boutillon, Christophe Jégo, and Michel Jézéquel, "An LDPC decoding method for fault-tolerant digital logic," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012.

[7] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[8] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 599–618, 2001.

[9] V. C. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, no. 3, pp. 299–301, 2003.

[10] C. Winstead, V. C. Gaudet, A. Rapley, and C. Schlegel, "Stocahstic iterative decoders," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2005.

[11] D. E. Muller and W. S. Bertky, "A theory of asynchronous circuits," in *Proc. International Symposium on the Theory of Switching, Part 1*, 1959, pp. 204–243.

[12] C. Winstead, "C-element multiplexing for fault-tolerant logic circuits," *Electronics Letters*, vol. 45, no. 19, pp. 969 –970, 10 2009.

[13] C. Winstead and M. El Hamoui, "Reducing clock jitter by using muller-c elements," *Electronics Letters*, vol. 45, no. 3, pp. 150 –151, 29 2009.

[14] Wenjing Rao, A. Orailoglu, and R. Karri, "Logic mapping in crossbar-based nanoarchitectures," *Design Test of Computers, IEEE*, vol. 26, no. 1, pp. 68–77, Jan.-Feb. 2009.