Majority Multiplexing: Economical Redundant Fault-Tolerant Designs for Nano Architectures

Sandip Roy, *Member*, *IEEE*, and Valeriu Beiu, *Senior Member*, *IEEE*
School of EE&CS, Washington State University, Pullman, WA 99164-2752
E-mail: {sroy, vbeiu}@eecs.wsu.edu

Abstract — **Motivated by the need for economical fault-tolerant designs for nano architectures, we explore a novel multiplexing-based redundant design scheme at small (≤ 100) and very small (≤ 10) redundancy factors. In particular, we adapt a strategy known as von Neumann multiplexing to circuits of majority gates with three inputs, and for the first time exactly analyze the performance of a multiplexing scheme for very small redundancies, using combinatorial arguments. We also develop an extension of von Neumann multiplexing that further improves performance by excluding unnecessary restorative stages in the computation. Our results show that the optimized three-input majority multiplexing (MAJ-3 MUX) outperforms the latest scheme presented in the literature, known as parallel restitution (PAR-REST), by a factor between two and four, for $48 \le R \le 100$. Our scheme performs extremely well at very small redundancies, for which our analysis is the only accurate one. Finally, we determine an upper bound on the maximum tolerable failure probability when any redundancy factor may be used. This bound clearly indicates the advantage of using three-input majority gates in terms of reliable operation.**

Index Terms — **Fault/defect tolerance, majority gates, nano architectures, von Neumann multiplexing.**

I. Introduction

The development of ever-smaller devices brings promise for further improvement in the performance of future integrated circuits, yet it also leads to several new technical challenges, including the need for nano architectures that reduce the uncertainty inherent to computation and communication at such small scales [1], [2]. One well-known approach for developing reliable architectures in the face of uncertainties, which include both *defects* in the fabricated chip and transient *faults* during operation, is to incorporate spatial and/or temporal *redundancy*. While redundancy is needed for reliable computation, however, economic constraints dictate that the *redundancy factor* R — the number of times a 'computing unit' is replicated — must be small or 'economical' ($R \leq 100$), or better very small or 'practical' ($R \leq 10$). This article presents a fresh view toward a thorough investigation of the feasibility of reliable architecture development using small and very small redundancy factors.

Fault- and defect-tolerant architectures have recently received revived attention in the nanotechnology community [3]–[10]. One can easily identify several reasons for further theoretical investigation of reliable redundant architectures in general, and of reliable redundant threshold logic gate (TLG) circuits in particular:

- *TLG circuits mimic the human brain, and hence hold promise in replicating the computational power of the brain*. While a TLG — which fires when some variable reaches a *threshold* — seems at first to be a drastic simplification of a neuron, benchmarking the four-dimensional neuron model of Hodgkin and Huxley against a TLG verifies the connection [11]. In particular, the TLG model was shown to correctly predict nearly 90% of the spikes generated by the Hodgkin and Huxley model in a stochastic simulation [11], thus *showing the considerable similarity between a neuron and a TLG*.

- Many theoretical results show that *TLG circuits are more powerful/efficient (in a computational sense) than classical Boolean circuits* (for reviews see [12]–[15]), motivating their further study.
- Theory also shows that *TLG circuits can be made arbitrarily fault-tolerant* (reliable) using a "small" amount of additional hardware (i.e., using a redundancy factor that is logarithmic in size), *while Boolean circuits cannot* [16].
- Finally — and most importantly — *nano-scale devices are unreliable*, and hence simulation and implementation of gates/circuits with unreliable components are of current interest (see [17], [18]).

All of these strongly motivate the following lines of investigation:

- Redundant design schemes [20]–[22] — e.g., modular redundancy, cascaded modular redundancy, multiplexing (MUX, including vN-MUX [19] and parallel restitution PAR-REST [10]), and reconfigurability [3], [8], [23] — *should be adapted to accommodate TLG circuits*.
- Redundant design schemes require further study *in the case of small (and hence economical) redundancy factors* ($R \leq 100$). Modifications and enhancements of these schemes (and of their analyses) that would allow extension to *very small redundancy factors* ($R \leq 10$) should be aggressively pursued, since these can significantly improve the tolerable failure probability in practical designs.
- *Paradigms for combining several redundancy schemes* should also be considered, in order to appropriately evaluate whether *very small* ($R \leq 10$) practical redundant designs can be reliable enough.

In this article, we present some new results concerning the first two of these directions of study.

Specifically, we modify vN-MUX for majority (MAJ) gates with 3 inputs (MAJ-3). We then develop an exact probabilistic analysis of our MAJ-3 MUX scheme using exact combinatorial arguments, which allows for the first time to accurately characterize a MUX scheme at small and very small redundancy factors. We are motivated to pursue an adaptation of vN-MUX, in particular, because it has shown promise for architectures with both high defect and fault probabilities [3]–[7], [10], and has been shown to achieve much better results than R-modular redundancy (R-MR) for very large redundancy factors ($R > 10{,}000$) [19]. Next, we develop an upper bound on the maximum tolerable gate failure probability $q_{\text{MAJ-3}}$, when any amount of redundancy may be used. This upper bound also underscores the advantage of using MAJ-3 gates rather than NAND-2 gates. Finally, we present an extension of vN-MUX that serves to further optimize its performance for very small redundancy factors ($R \leq 10$).

The remainder of the article is organized as follows. In Section II, we describe briefly reliability-related work from the literature. In Section III, we characterize MAJ-3 MUX, and do a probabilistic analysis of a single multiplexed computation using worst-case exact combinatorial arguments. We also present an upper bound on the failure probability p_f, when any redundancy factor R may be used. In Section IV, we use our analysis to compare the required device-failure probabilities p_f for achieving chip-level reliability, given that MAJ-3 MUX at a particular redundancy factor R is implemented. In Section V, an extension of the multiplexing method is presented, and shown to further improve reliability significantly for very small redundancy factors. In Section VI, we summarize our findings and briefly discuss directions for further research.

II. Literature Review

Reliable operation of a circuit can be achieved using redundancy at many different levels: at the device level [24], [25]; at the gate level [26]; at the block level [27]; in time; and in communication (through encoding, e.g., [28]) (see also [3]–[10] and [17]). While most of these methods are beyond our scope in this article, we note that they have in common that improved reliability is traded off for increased chip area and higher connectivity [29], [30]. In this paper, we only focus on redundant designs at the gate level.

In the circuit complexity community, fault/defect-tolerance at the gate level has been formalized under two models: *the gate error model* where only gates are assumed to be faulty, and *the wire error model* where only wires make errors [16]. The wire fault model tends to be appropriate for circuits with complex gates because, for instance, it can capture that a gate with a larger fan-in might have a larger probability of error. The fan-ins of the gates we analyze in this article are very limited (*fan-in* = 3), and hence the gate error model is quite appropriate. That is, we assume that only gates can cause errors, and that there is a fixed upper bound on their error probability, regardless of the fan-in of the gate. To prevent amplification of these errors in a logic circuit, some type of redundant design must be used. We will quantify redundancy using the redundancy factor R, which indicates the multiplicative increase in circuit size (i.e., number of gates) required to attain fault-free operation, or equivalently the ratio of the size of the fault/defect-tolerant circuit to the size required in case of no faults. [*Remark: In [29] and [30] Reischuk and Schmeltz aptly point out that increase in circuit area rather than increase in circuit size is a more significant measure of redundancy, and show how encoding in combination with replication can be used to minimize circuit area.*]

In [19] von Neumann introduced the multiplexing redundancy algorithm vN-MUX as a plausible representation for reliable computation. vN-MUX was developed for arbitrary gates, including MAJ and NAND (known then as Sheffer Stroke) gates (see Fig. 1 showing the executive stage followed by two restorative stages). However, a detailed reliability analysis was performed for two-input NAND (NAND-2) gates only, assuming independent gate failures and very large redundancy factors. By exploiting the approximate Gaussianity of the number of failures in the executive stage output, von Neumann showed that long sequences of computations could be performed reliably for sufficiently small gate failure probability. His analysis yielded a maximum tolerable failure probability per NAND-2 gate of $q_{\text{NAND-2}} = 0.0107$ for sufficiently large redundancy. A detailed explanation of this result was recently given in [10].

The performance of NAND-2 vN-MUX was compared with the performance of other fault tolerance techniques in [3]–[7] (see Fig. 2). It was shown that NAND-2 vN-MUX can accommodate devices with a defect probability $p_f \leq 0.01$. However, this maximum defect probability is only achievable at the expense of enormous redundancy factors (R ~ 1,000,000).

In [7], NAND-2 vN-MUX was analyzed at small to moderate redundancy factors of 30, 300, and 3000. It was shown that the number of faulty outputs from a single multiplexed logic computation is binomial for these redundancies, and can be approximated as Gaussian *only for moderate to large redundancy factors* ($R > 3000$). NAND-2 vN-MUX has been analyzed using a CAD tool in [31]. The results reported in [31] show that for small redundancy factors the theoretical results from [7] are inaccurate, while PRISM [32], [33] is able to provide more precise estimates. The framework for PRISM is based on probabilistic model checking, and can be used for evaluation of the reliability/redundancy trade-off for various designs. In their analysis [31], the authors reported simulation results for small redundancy factors, i.e., 60, 120, and 180,

for the NAND-2 vN-MUX scheme. Another simulation tool for reliability calculations was recently described in [34]. It comprises MATLAB-based libraries for fundamental logic gates that can compute output probability distributions based on the input distribution. This approach is based on the analysis of Bahar et al. [35], where a Boolean gate is modeled using an energy distribution function. Another tool dedicated for estimating the reliability of quantum cellular automata (QCA) is presented in [36]. It can be integrated with QCADesigner, a tool for the layout and simulation of QCA structures. The approach suggested for increasing redundancy is a low-level one, based on triplicating both gates (MAJ) and wires. Very recently [28], examples of hardware architectures that incorporate one or multiple redundancy schemes (triple modular redundancy together with encoding) were tested using VHDL/Spice/Monte Carlo simulations.

The PAR-REST scheme [10] is of particular interest. While PAR-REST has many features similar to vN-MUX, the authors distinguish between the schemes based on the fact that the computations are not collapsed after each layer of the circuit (see [10] for details) and that *restorative stages* (defined later) are only used periodically. The article [10] shows that PAR-REST can significantly improve upon NAND-2 vN-MUX for small to moderate R. Our independent studies in the earlier works [37], [38], and in this article, also do not collapse the result from each multiplexing stage and propose periodic restoration, though we tend to view this modification as a modification of vN-MUX rather than a wholly new scheme.

Novel redundancy techniques that combine device-level and gate-level design ideas have also been introduced. In [25] the authors propose a redundant design approach that creates a rescaled weighted average of the redundant blocks' outputs. This results in a multiple-valued logic representation (of the function to be implemented), and provides an effective means of absorbing faults. The authors show, through several examples, that the new design technique improves the

immunity to permanent and transient faults occurring *at the transistor level*, and works even for a redundancy factor $R = 2$. The paper suggests that dynamically adjustable threshold levels may further enhance this method. Another approach proposes a robust design of Boolean gates as feedforward artificial neural networks of small size, specifically trained to absorb errors [26]. After learning, such gates could be incorporated in a library of fault tolerant gates, thus becoming transparent to the VLSI designer (if the synthesis tools could take advantage of them). The differential configuration used can also improve on the robustness of data transmission, but at the expense of additional power consumption and increased switching activity (noise). The authors mention that this method might be difficult to implement due to the high precision required for the weights. Other low-level (i.e. device/gate) approaches which we should mention here belong to the larger class of rad-hard by design [39], and high matching techniques used in analog circuits [40], [41] (recently used for enhancing the reliability of CMOS TLGs [24]).

The methods for reliable design of nano architectures that we have described generally adhere to the following philosophy: a particular redundancy scheme is chosen, and the reliability of a typical class of circuits is evaluated given this redundancy scheme. A second body of literature on reliability seeks to analyze the minimum redundancy required to implement a particular faulty computation, assuming that *any* redundancy scheme may be used [16], [29], [30], [42]–[53]. Broadly speaking, these studies seek to construct a reliable architecture of minimum size for a particular logic function, given the failure probability of gates and/or wires. This literature is largely beyond this article's scope, so we shall present only a few relevant results. The maximum tolerable failure probability for any redundant NAND-2 circuit was shown to be $q = 0.08856$ in [51], but the suggested redundant design in this case is not vN-MUX. A similar analysis has been performed recently for TLG circuits, in which case gate failure probabilities up to $q = 0.25$ can be

tolerated [16]. In particular, Reischuk investigated the required redundancy for circuits with unbounded fan-in. He concluded that TLG circuits have a clear advantage over Boolean gate circuits with respect to reliability (fault/defect-tolerance). More precisely, *only* using unbounded TLGs can arbitrary circuits be made reliable (fault/defect-tolerance) with a small amount of additional hardware (logarithmic in size). Boolean circuits of unlimited fan-in *cannot* be made reliable even if large redundancy factors are allowed. Finally, the error threshold of the gate represents another advantage of MAJ over NAND, as can be seen in Fig. 3. In this figure, the error threshold for MAJ-k gates (MAJ gates of fan-in = k) is $\varepsilon_{MAJ}(k) = \frac{1}{2} - \frac{2^{k-2}}{k\binom{k-1}{(k-1)/2}}$ for k odd (as determined in [49], [53]), while the error threshold ε_{NAND} for NAND-k gates was computed by numerically solving $\left(1+\frac{1}{k}\right)\left[\frac{1}{k(1-2\varepsilon_{NAND})}\right]^{1/(k-1)} = 1-\varepsilon_{NAND}$ (as recently proven in [54]).

III. Multiplexing for a Single MAJ-3 Computation

A. Description

The vN-MUX algorithm developed in [19] aims to improve the reliability of a sequence of logic computations, by assimilating the results from redundant implementations of each computation in the sequence. This 'multiplexing' of each computation serves to contain error propagation down the sequence, by selecting for the more common (and hence more-likely correct) result at each stage. The vN-MUX scheme can be applied to any logic computation, but the analysis of the scheme must be adapted for each case.

A single MAJ-3 vN-MUX logic computation is presented in Fig. 4. The vN-MUX computation comprises an *executive stage* and a *restorative stage*. The executive stage repeats

the desired logic computation (in our case a MAJ-3 computation) a total of N times, operating on N different sets of inputs obtained from the previous computation. The restorative stage triplicates and randomly orders (see randomizer in Fig. 4(b)) the outputs from the executive stage, and then chooses the majority of each randomly-chosen set of three signals using a second set of N MAJ-3 gates, to generate the N final outputs. The purpose of the restorative stage is to increase the frequency of the more common output from the executive stage. This restoration is central to the global performance of the vN-MUX scheme. It is important to note that we do not collapse the N different outputs into a single one, and so we use the term MAJ-3 MUX rather than MAJ-3 vN-MUX hereafter.

B. Probabilistic Analysis

Our aim is to relate the output error probabilities of a MAJ-3 MUX computation to the input error probabilities and the MAJ-3 gate failure probabilities $q_{\text{MAJ-3}}$. Our statistical model for this MAJ-3 MUX computation is as follows. We assume that each of the three sets of N inputs — henceforth called a bundle as in the original article [19] — has a nominal binary value. Let the number of lines in bundle i, $1 \leq i \leq 3$, that are in error (i.e., deviate from their nominal input value) be denoted e_i. Given e_i, all configurations of line errors among the N lines in bundle i are assumed to be equally probable. The three random variables e_1, e_2, and e_3 are assumed to be independently distributed, with probability distributions $P(e_1)$, $P(e_2)$, and $P(e_3)$, respectively. We also assume that each MAJ-3 gate can fail with probability $q_{\text{MAJ-3}}$. Based on the nominal inputs, we can determine the nominal final output. We define d as the number of final outputs (outputs from the restorative stage) that differ from the nominal output. Our goal is to characterize the distribution of d.

The distribution of d depends on the values of the nominal inputs — in particular, on whether or not there is a consensus among the nominal inputs. It is easy to see that the probability of malfunction (i.e., the probability that a significant number of outputs differ from the nominal output) is greatest when the nominal inputs are not in consensus. Since our concern is about the reliability of this design, *we will evaluate the distribution of d in the worst case*, i.e. when the inputs are not in consensus. Without loss of generality, let us assume that that inputs I_1 and I_2 have a nominal value of *high*, while input I_3 has a nominal value of *low*. Thus, the nominal output is *high*. Also, we note that the probability of malfunction decreases as the number of errors in the single distinct input (input I_3 in our case) increases. In our calculations, we thus assume the worst case, i.e. that input I_3 is always correct, and so e_3 is always *low*.

We will require several steps to compute the distribution of d. First, we define a variable w for the number of MAJ-3 gates in the executive stage that have at least two inputs *high*. To find the distribution for w, let us first find the conditional distribution for w given e_1 and e_2, or $P(w \mid e_1, e_2)$. This conditional distribution can be found using a counting argument, as follows. In total, there are $\binom{N}{e_1} \cdot \binom{N}{e_2}$ configurations for the line errors in all three input bundles (recall that the third set of lines is assumed to have no errors). Each of these configurations has w MAJ-3 gates with at least two inputs *high* if and only if neither line 1 (i.e., the line from bundle 1) nor line 2 has errors at w MAJ-3 gates, and either line 1 or line 2 has an error at the other $N-w$ MAJ-3 gates. (Note that, in consequence, the probability is zero unless $w \le \min(e_1, e_2)$ and $e_1 + e_2 \ge N - w$.) The number of ways in which we can choose w MAJ-3 gates, for which both input I_1 and input I_2 are *high*, is $\binom{N}{w}$. The number of ways in which to distribute the errors

among the lines to the remaining $N-w$ MAJ-3 gates, so that either input I_1 or input I_2 has an error, is:

$$\binom{N-w}{e_1}\cdot\binom{e_1}{N-e_2-w}. \tag{1}$$

Hence, we find that the conditional probability for w given e_1 and e_2 is:

$$P(w\,|\,e_1,e_2)=\frac{\binom{N}{w}\cdot\binom{N-w}{e_1}\cdot\binom{e_1}{N-e_2-w}}{\binom{N}{e_1}\cdot\binom{N}{e_2}}. \tag{2}$$

Finally, we can compute the probability distribution for w as:

$$P(w)=\sum_{e_1=1}^{N}\sum_{e_2=1}^{N}P(w\,|\,e_1,e_2)\cdot P(e_1)\cdot P(e_2). \tag{3}$$

We note that the combinatorial arguments used to find the distribution of w are similar to those given in [19], for analysis of the executive stage of a NAND-2 computation.

Once we have computed the distribution for w, we can compute the distribution for s — the number of correct outputs (i.e., the number of outputs that are *high*) after the executive stage. An output from a MAJ-3 gate is guaranteed to be correct only if at least two of its inputs are *high*, and the MAJ-3 gate is functioning. Hence, the conditional distribution for s given w is binomial: $P(s\,|\,w)=\binom{w}{s}\cdot(1-q_{\text{MAJ}-3})^{s}\cdot q_{\text{MAJ}-3}^{w-s}$, and the (unconditioned) distribution for s is $P(s)=\sum_{w=s}^{N}P(s\,|\,w)\cdot P(w)$.

The next signal that we consider is a, the number of MAJ-3 gates in the restorative stage that have at least two inputs *high*. Note that the inputs into the restorative stage MAJ-3 gates are generated from the outputs of the executive stage, through triplication and randomization. To

compute the probability distribution for a, we first compute the conditional distribution for a given s, using a combinatorial argument. In particular, we determine the number of distinct input arrangements (consisting of $3N$ inputs) into the N MAJ-3 gates that make up the restorative stage, and count the number of such arrangements for which a MAJ-3 gates have at least two inputs *high*. Given s, $3s$ of the $3N$ inputs into the restorative stage are *high*. Hence, the *high* inputs of the MAJ-3 gates in the restorative stage can be arranged in $\binom{3N}{3s}$ different ways, each of which is equally probable (because of the action of the randomizer). Next, we aim to count the number of such sequences for which exactly a MAJ-3 gates have at least two inputs *high*. We will call these the *good* sequences. To count the number of good sequences, we count the number of good sequences that have exactly i_3 gates with all three inputs *high*, and add over all i_3. Note that $i_3 \leq \min(s, a)$. For a particular i_3, the number of MAJ-3 gates with exactly two inputs *high* is $i_2 = a - i_3$, since a MAJ-3 gates have at least two inputs *high*. Next, since the total number of *high* inputs is $3s$, the total number of MAJ-3 gates i_1 with exactly one input *high* must be $i_1 = 3s - 3i_3 - 2i_2$, where it is required that $i_1 \geq 0$. The number of MAJ-3 gates i_0 with no inputs *high* must be $i_0 = N - i_3 - i_2 - i_1$, where i_0 is further constrained to be non-negative. Since i_3 determines i_2, i_1, and i_0, the number of good sequences having a particular i_3 is the number of good sequences with the resultant i_3, i_2, i_1, and i_0. To count this number, we note that MAJ-3 gates with three, two, one, and zero inputs *high* can be arranged arbitrarily, and further that the inputs into each MAJ-3 gate with exactly one or two inputs *high* can be arranged in three different ways (100, 010, 001, and respectively 110, 101, 011). Hence, the number good sequences with a particular i_3 is given by

$$\binom{N}{i_3}\cdot\binom{N-i_3}{i_2}\cdot\binom{N-i_3-i_2}{i_1}\cdot 3^{i_2}\cdot 3^{i_1}$$

$$=\binom{N}{i_3}\cdot\binom{N-i_3}{a-i_3}\cdot\binom{N-a}{3s-3i_3-2(a-i_3)}\cdot 3^{a-i_3}\cdot 3^{3s-3i_3-2(a-i_3)}, \tag{4}$$

where a combination is assumed to be zero if its lower term is smaller than zero, or greater than the upper term. It follows from (4) that the conditional probability for a given s is

$$P(a\mid s)=\frac{\sum_{i_3=0}^{\min(a,s)}\binom{N}{i_3}\cdot\binom{N-i_3}{a-i_3}\cdot\binom{N-a}{3s-3i_3-2(a-i_3)}\cdot 3^{a-i_3}\cdot 3^{3s-3i_3-2(a-i_3)}}{\binom{3N}{3s}}. \tag{5}$$

As we have done before, we can compute the unconditioned distribution for a by finding the joint distribution of a and s, and marginalizing.

We can compute the distribution of the number of output errors d from the distribution of a, by considering failures in the restorative-stage MAJ-3 gates. A binomial distribution represents the conditional distribution in this case. We omit the details, since this analysis is identical to our analysis of the executive stage presented before.

In his seminal paper [19], von Neumann also observed that accurate (essentially error-free) computations are possible using NAND-2 vN-MUX only if the probability of a NAND-2 gate failure probability $q_{\text{NAND-2}}$ is below a certain threshold value. In particular, if $q_{\text{NAND-2}}$ is below this threshold value, the frequency of errors at the output of a multiplexing stage is less than or equal to the frequency of errors at the input of that stage — given that a sufficiently large bundle size N is used. Thus, a sequence of computations can be completed without increasing the error frequency. We have observed a similar behavior for MAJ-3 MUX. In particular, we have determined that MAJ-3 MUX can achieve accurate computations for gate failure probabilities

$q_{\text{MAJ-3}} < 0.0197$. This outperforms the NAND-2 gate failure probabilities $q_{\text{NAND-2}} < 0.0107$. We have obtained this threshold in an analogous manner to [19]: by invoking the central limit theorem, we identify a deterministic equivalence for the error frequency update, in the limit of large bundle size. The deterministic equivalence allows us to compute whether the error frequency decreases over a computation, for a given gate failure probability. A more detailed discussion of this threshold result will be presented in future work, while a relevant discussion for NAND-2 gates can be found in [10].

IV. Chip-Level Analysis

In the previous section, we showed how to obtain the probability distribution of the number of *high* outputs of a single MAJ-3 MUX computation given the MAJ-3 gate failure probability $q_{\text{MAJ-3}}$. In this section we extend the analysis to the chip level, by going from a device to a gate, and finally to the system (chip). These steps will allow us to determine the relationship between the device failure rate (p_f) and the chip failure rate (P_{chip}). Our chip-level analysis follows the analysis of [3]–[5], [7].

The first step is represented by the device failure rate p_f. This is an elusive measure, which for current CMOS processes can be estimated to be in the 10^{-7} to 10^{-8} range, but will be adversely affected by scaling (see also [55], [56]). For example, for SET the expectations are that about one in one thousand devices might have considerable background charge fluctuations, i.e. $p_f \approx 10^{-3}$ [57].

The second step is to determine the gate failure rate $q_{\text{MAJ-3}}$ in terms of the device failure rate p_f. A simple equation relating the two is $q_{\text{MAJ-3}} = 1 - (1 - p_f)^j$, where j is the number of devices in a MAJ-3 gate (see [3], [4]). This simplified model does not take into account the fact that

different devices (i.e., both active ones like MOS transistors, SET junctions, RTDs, etc., and passive ones like resistors and capacitors) might have different failure probabilities. Also, this model does not consider connections, i.e., both wires (which for a CMOS process can be on several different layers, and hence might have different failure probabilities) and contacts (between the different layers). A conventional CMOS implementation requires 4 transistors (2 nMOS and 2 pMOS) for a NAND-2 gate, giving $q_{\text{NAND-2}} = 1 - (1 - p_f)^4$ [3], [4]. A standard domino implementation of a NAND-2 also requires 4 transistors (a solution with 3 transistors is possible). It was suggested that other Boolean gates could be replaced by NAND-2 gates [3], [4]. The authors also mention that a MAJ-3 gate requires 4 NAND-2 gates, and conclude that the use of NAND-2 gates is in the interest of minimizing redundancy. While this is true if MAJ-3 gates are implemented using NAND-2 gates, there are other CMOS—and also SET, RTD, and molecular—designs which could be used for implementing MAJ-3 gates [18]. The simplest CMOS solution is a pseudo-nMOS implementation requiring 4 transistors: 3 nMOS for the 3 inputs, and 1 pMOS acting as load. This gives $q_{\text{MAJ-3}} = 1 - (1 - p_f)^4$, suggesting that a MAJ-3 (pseudo-nMOS) gate might be as reliable as a NAND-2 (CMOS) gate. In fact things are much more complicated, and many other designs are possible, which although having more devices can have a lower probability of failure. For example, using high matching techniques ([40], [41]), the number of devices is increased, but because several (identical but smaller) devices are connected in parallel, they build redundancy at the lowest possible level (i.e., the device level), and so increase the overall reliability of the gate [24], [58].

The third step assumes that the chip has N_{TOTAL} devices, and is divided into processing units of effective device count N_{unit}. Since the MAJ-3 MUX introduces a redundancy factor of $R = 2 \times N$ (we note here that the NAND-2 vN-MUX has a slightly larger $R = 3 \times N$), the number of

processing units is given by $n = N_{TOTAL} / (R \times N_{unit}) = N_{TOTAL} / (2 \times N \times N_{unit})$. We also assume that the units have a logical depth $D = 10$, as has been done previously in [3]–[5], [7]. We thus apply the probability analysis of Section III to the logical units sequentially to obtain the final probability distribution $P_D(d) = P_{10}(d)$. We then assume a critical threshold level $\theta = 0.5$, and calculate the probability that $d/N \geq \theta = 0.5$. [*Remark: We note here that von Neumann's original study [19], as well as some more recent studies [3]–[7], use thresholds smaller than 0.5. Such thresholds are needed to maximize the tolerable probability of error p_f when a decision is taken at each multiplexing stage. We advocate taking a decision (voting in our case) only after the final stage, and so a threshold of 0.5 suffices.*] This is the probability that one data bit (P_{bit}) is wrong. We note that the implementation of the threshold at the output requires additional logic circuitry. As has been done in the literature ([3]–[10] and [19]), we assume that the failure probability of the final thresholding circuit is negligible. Very high reliability logic gates can be designed in practice by introducing redundancy at the device level [24]–[26], [36], [39]–[41], [58].

The reliability of one processing unit is thus given by $P_{unit} = (P_{bit})^m$, where m is the number of outputs from a processing unit. Hence, the reliability of the whole chip is $P_{chip} = (P_{unit})^n$. Putting all of these together we get

$$P_{chip} = (P_{unit})^n = [(P_{bit})^m]^n = \{[P_D(d)]^m\}^n = \{[P_{10}(d)]^m\}^{N_{TOTAL}/(R \times N_{unit})}. \quad (6)$$

Now that we have related the chip reliability P_{chip}, i.e. the probability that the chip produces a correct output *during a single clock cycle*, to the device failure rate p_f for a given redundancy factor R, we can determine the maximum allowed failure probability p_f as a function of R for given P_{chip}, m, N_{TOTAL}, and N_{unit}. We consider a chip with $N_{TOTAL} = 10^{12}$ devices and $N_{unit} = 10^6$ processing units, each of which returns a single bit $m = 1$. Through simulation, we determine the

maximum failure probability p_f needed to guarantee a chip reliability of $P_{chip} = 90\%$, as a function of the redundancy factor R. Our results for small redundancy factors ($R < 100$) are shown in Fig. 5. We note that these results provide a significant improvement over R-MR and NAND-2 vN-MUX [3]–[7], and stress again that our results are the only ones that are accurate at very small redundancies.

A comparison of greater interest is between MAJ-3 MUX and PAR-REST [10], since PAR-REST constitutes the latest scheme recently proposed in the literature. The article [10] considers reliability over a duration of time (i.e., multiple clock cycles), and reports maximum failure probabilities when 90% reliability over ten years of processing is demanded. We compare MAJ-3 MUX with PAR-REST using this reliability demand and identical chip specifications as in [10]. Because PAR-REST is most fairly compared with a scheme that only periodically uses restoration, we delay this comparison to the next section. Finally, we note that we are currently investigating a more realistic chip-level analysis that meshes multiple redundancy strategies, including device level ones, and will report on the results in a subsequent article.

V. An Extension of von Neumann Multiplexing

The purpose of the restorative stage in MUX is to reduce error propagation from a logic computation's input to its output, by selecting for the more common outputs from the computation. The restorative stage is only effective when the probabilities of error in the inputs are sufficiently large. In fact, for small input error probabilities, the chance of error introduced by the gates in the restorative stage might outweigh the advantage of having the restorative stage. Thus, if the input error probabilities, for a particular logic computation, are small enough, we can

simultaneously improve the output error probability and economize (reduce the redundancy factor R) of any MUX design by eliminating the restorative stage.

From another viewpoint, if we are seeking the best-performing architecture for a particular redundancy factor R, we might expect to *improve on the standard vN-MUX algorithm by applying the restorative stage on only some computations, while simultaneously increasing the bundle size N*. For instance, say that we wish to design a MUX architecture with a maximum redundancy factor $R_{\text{MAX}} = 15$. A classical NAND-2 vN-MUX would use a bundle size of $N = 5$, and apply restoration on every computation, hence $R_{\text{NAND-2}} = 3\times5 = 15 = R_{\text{MAX}}$. If we were to apply the standard MAJ-3 MUX, we would use a bundle size of $N = 7$ and apply restoration on every computation, which yields $R_{\text{MAJ-3}} = 2\times7 = 14 < R_{\text{MAX}}$. It turns out that even the reliability of this architecture can be improved by using a bundle size of $N = 11$ and applying restoration only every third computation (to be precise, for computations at depths 3, 6, and 9 in the logic circuit). This design yields an 'average' redundancy factor of $R_{Enhanced_\text{MAJ-3}} = 11 + 11/3 = 14.3 < R_{\text{MAX}}$. We use this idea to improve our MAJ-3 MUX, but the same principle can be applied when using any other type of gate or combinatorial logic block. Specifically, we consider architectures in which the logical depths of all inputs to a given computation are the same (but in general this need not be the case). We define *the enhanced MAJ-3 MUX(N, k) architecture* as one in which an executive stage with bundle size N is used for all computations, and a restorative stage is applied only on every kth stage (i.e., for computations with logical depth k, $2k$, … — *while in general the restorative stages could be distributed unevenly*). We note that the redundancy factor introduced by a MAJ-3 MUX(N, k) architecture is $R = N + N/k$. By placing the restorative stage only every kth stage, the bundle size N can be increased to $N^+ = [2k/(k+1)]\times N$ for the same redundancy factor R.

We have computed the maximum probability of failure for a device p_f, such that a reliable chip-level design $P_{chip} = 0.9$ can be achieved for a given redundancy factor R (see Fig. 5). However, we now consider all MAJ-3 MUX(N, k) architectures, i.e. all possible (N, k) pairs satisfying a given R. We find this probability by computing the maximum allowed probability of device failure for each architecture for the given R, and then choosing the architecture that maximizes the allowed failure probability. For instance, for a redundancy factor $R = 15$, we found that the MAJ-3 MUX(11, 3) architecture provides the maximum allowed probability of device failure, of 6.25×10^{-5}. Fig. 6 shows the probabilities of failure allowed for each redundancy factor using the best possible MAJ-3 MUX(N, k) architecture. To illustrate our enhanced MAJ-3 MUX(N, k) scheme, we present a detailed comparison of MAJ-3 MUX(N, k) with MAJ-3 MUX in Fig. 7. It shows that:

- for small but not very small redundancy factors, $10 < R < 100$, MAJ-3 MUX(N, k) does not significantly improve on MAJ-3 MUX (see Fig. 7(a));
- on the other hand, for very small redundancy factors, $R < 10$, MAJ-3 MUX(N, k) architectures provide an additional four orders of magnitude over MAJ-3 MUX (see Fig. 7(b)), with the strongest advantages at $R = 3$, 4, and 5.
- MAJ-3 MUX(N, k) seems to be always better than R-MR and is quite able to compete with reconfiguration ([3], [8], [23]).

An exact numerical comparison for very small redundancy factors can be seen in Table I. The table also presents the particular (N, k) that achieves the best performance. An interesting additional advantage of using MAJ-3 MUX(N, k) is that we can tailor the designs to achieve desired redundancy factors.

Finally, we return to the comparison between MAJ-3 MUX and PAR-REST. As in enhanced MAJ-3 MUX*(N,k)*, PAR-REST takes advantage of periodic restoration to improve performance. Hence, a comparison between MAJ-3 MUX and PAR-REST is perhaps the most fair comparison of the reliability of MAJ-3 and NAND-2 architectures. We have compared the performance of MAJ-3 MUX and PAR-REST at the smallest analyzed redundancy for PAR-REST ($R = 48$) and also at $R = 100$ (which is the largest redundancy we have considered for MAJ-3 MUX). At $R = 48$, MAJ-3 MUX improves on PAR-REST by a factor of 1.5 (i.e., 2.3×10^{-4} versus about 1.5×10^{-4}), while achieving a factor of 4.25 at $R = 100$ (i.e., 1.7×10^{-3} versus about 4×10^{-4}).

VI. Conclusions and Future Directions of Research

In this article, we have presented an engineering analysis of multiplexing. We have developed the first exact and worst case analysis for MAJ-3 MUX for small and very small redundancy factors, using combinatorial arguments. We have also presented an extension of MAJ-3 MUX that can significantly improve its performance at very small redundancy factors ($R < 10$). We have compared these schemes to other redundancy schemes, showing that MAJ-3 MUX holds promise for economical (i.e., small) as well as practical (i.e., very small) redundant fault-tolerant designs. In fact, MAJ-3 MUX improves on the performance of R-MR by five orders of magnitude, and by about four orders of magnitude over classical NAND-2 vN-MUX. An additional three to four orders of magnitude can be gained at very small redundancy factors ($R = 3, 4, 5$) when using enhanced MAJ-3 MUX(N, k), a scheme that we introduced here. These comparisons show that MAJ-3 MUX gives better results than R-MR for very small redundancy factors ($R < 10$), and is able to compete with reconfiguration for small redundancy factors

($R < 100$). The enhanced MAJ-3 vN-MUX(N, k) outperforms R-MR for any $R > 2$, and starts competing with reconfiguration even at very small redundancy factors ($R < 10$).

In this article, we have also presented an upper bound on the maximum tolerable failure probability, indicating the clear advantage of using MAJ-3 gates over NAND-2 gates in multiplexing.

Our future work will focus on several lines of research. The first one is to investigate combinations of MAJ-3 MUX(N, k) with low-level reliable schemes. This will allow for additional (hopefully significant) reductions in area. As an example, the MAJ-3 MUX(3, ∞) (see Fig. 8(a)) has been used in combination with a high-matching inspired technique [40], [41] (see [24] for a CMOS application to TLGs). These were used together to improve the robustness of capacitive SET TLGs with respect to process variations [58]. Data was collected from 10,000 simulations using SIMON (a Monte Carlo SET simulator) and our own MATLAB modules. Both the probability of failure of one MAJ-3 SET gate $q_{\text{MAJ-3}}$ versus process variations, as well as the probability of failure of a MAJ-3 MUX(3, ∞) structure (see Fig. 9(a)), implemented using high matching MAJ-3 SET gates (see Fig. 9(b)), can be seen in Fig. 8(b). This figure shows that while one MAJ-3 SET gate starts giving errors from $\pm 3\%$ variations (on all junctions and capacitor values), the enhanced MAJ-3 vN-MUX scheme using high-matching MAJ-3 SET gates starts to err only from $\pm 5\%$ variations. It also can be seen that the robustness (tolerance) to variations is improved by over three orders of magnitude up to $\pm 6\%$ variations, by over two orders of magnitude for variations in the range $\pm 7\%$ to $\pm 8\%$, and over one order of magnitude for variations up to $\pm 10\%$. The same structure is currently being characterized in 120 nm standard CMOS (in subthreshold using an original adaptive body bias technique [59].

We are also pursuing further improvement and analysis of our MAJ-Δ MUX(N, k), and plan to present these developments in a subsequent article. It is known [49], [53], [54], that gates with larger fan-ins are expected to give an additional improvement (see Fig. 3). Thus, we plan to first extend MAJ-3 MUX to MAJ-5 MUX, as a step towards finding optimal MAJ-Δ MUX(N, k) structures. The extension from MAJ-Δ to arbitrary TLGs with very small delay penalty was theoretically proven in [60], [61]. Other techniques for improving performance that we are also following include unevenly sequencing the computations with and without restoration, and dynamically varying (re-sizing) the bundle size N from one stage to another.

REFERENCES

[1] *International Technology Roadmap for Semiconductors*, ITRS, 2003. Available: http://public.itrs.net/

[2] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, Jul.-Aug. 2003, pp. 14–19.

[3] M. Forshaw, K. Nikolić, and A. S. Sadek, "ANSWERS: Autonomous Nanoelectronic Systems With Extended Replication and Signaling," Microelectronics Advance Research Initiative (MEL-ARI) #28667, 3rd Year Annual Report, 2001, pp. 1–32. Available: http://ipga.phys.ucl.ac.uk/research/answers/reports/3rd_year_UCL.pdf

[4] K. Nicolić, A. S. Sadek, and M. Forshaw, "Architectures for reliable computing with unreliable nanodevices," *Proc. IEEE-NANO'01*, Maui, HI, USA, Oct. 28-30, 2001, pp. 254–259.

[5] K. Nikolić, A. S. Sadek, and M. Forshaw, "Fault-tolerant techniques for nanocomputers," *Nanotechnology*, vol. 13, no. 3, Jun. 2002, pp. 357–362.

[6] J. Han, and P. Jonker, “A fault-tolerant technique for nanocomputers: NAND multiplexing,” *Proc. Annual Conf. of the Advanced School for Computing and Imaging ASCI’02*, Lochem, The Netherlands, Jun. 19-21, 2002, pp. 59–66.

[7] J. Han, and P. Jonker, “A system architecture solution for unreliable nanoelectronic devices,” *IEEE Trans. Nanotech.*, vol. 1, no. 4, Dec. 2002, pp. 201–208.

[8] J. Han, and P. Jonker, “A defect- and fault-tolerant architecture for nanocomputers,” *Nanotechnology*, vol. 14, no. 2, Feb. 2003, pp. 224–230.

[9] J. Han, and P. Jonker, “A study on fault-tolerant circuits using redundancy,” *Proc. VLSI’03 (Multiconference in Comp. Sci. and Eng.)*, Las Vegas, NV, USA, Jun. 23-26, 2003, pp. 65–69.

[10] A. S. Sadek, K. Nikolić, and M. Forshaw, “Parallel information and computation with restitution for noise-tolerant nanoscale logic networks,” *Nanotechnology*, vol. 15, no. 1, Jan. 2004, pp. 192–210.

[11] W. M. Kistler, W. Gerstner, and J. L. van Hemmen, “Reduction of the Hodgkin-Huxley equations to a single-variable threshold model,” *Neural Computation*, vol. 9, no. 5, Jul. 1997, pp. 1015–1045.

[12] F. Scarselli, and A. C. Tsoi, “Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results,” *Neural Networks*, vol. 11, no. 1, Jan. 1998, pp. 15–37.

[13] V. Beiu, “A survey of perceptron circuit complexity results,” *Intl. Joint Conf. Neural Networks IJCNN’03*, vol. 2, Jul. 2003, pp. 989–994.

[14] J. Šíma, and P. Orponen, “General-purpose computation with neural networks: A survey of complexity theoretic results,” *Neural Computations*, vol. 15, no. 12, Dec. 2003, pp. 2727-2778.

[15] M. Anthony, “Boolean functions and artificial neural networks,” *Tech. Rep. LSE-CDAM-2003-01*, Department of Mathematics, London School of Economics, Jan. 2003. Available: http://www.cdam.lse.ac.uk/Reports/Files/cdam-2003-01.pdf
To appear in Y. Crama, and P. L. Hammer (eds.): *Boolean Functions: Theory, Algorithms, and Applications* (book in progress).
Available: http://www.sig.egss.ulg.ac.be/rogp/Crama/Publications/BookPage.html

[16] R. Reischuk, “Can large fanin circuits perform reliable computations in the presence of faults?,” *Theoretical Comp. Sci.*, vol 240, no. 12, Jun. 2000, pp. 319–335. Preliminary version in *Proc. Intl. Computing and Combinatorics Conf. COCOON’97*, Shanghai, China, Aug. 20-22, 1997, pp. 72–81. Also as *Tech. Rep. SIIM-TR-97-05*, Inst. for Theoretical Comp. Sci., University of Lübeck, Lübeck, Germany.
Available: http://www.itheoi.mu-luebeck.de/Forschung/A9705b.ps

[17] R. Compañó, L. Molenkamp, and D.J. Paul (Eds.), *Technology Roadmap for Nanoelectronics*, European Commission, IST Programme, Future and Emerging Technologies, Microelectronics Advanced Research Initiative (MEL-ARI NANO), 2000.
Available: http://www.cordis.lu/esprit/src/melna-rm.htm

[18] V. Beiu, J. M. Quintana, and M. J. Avedillo, “VLSI implementation of threshold logic: A comprehensive survey,” *IEEE Trans. Neural Networks* (*Special Issue on Hardware Implementations*), vol. 14, Sep. 2003, pp. 1217–1243.

[19] J. von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” in C. E. Shannon, and J. McCarthy (Eds.), *Automata Studies,* Princeton, NJ: Princeton Univ. Press, 1956, pp. 43–98.

[20] B. W. Johnson, *Design and Analysis of Fault Tolerant Digital Systems*, New York, Addison-Wesley, 1989.

[21] D. P. Siewiorek, and R. S. Swarz, *Reliable Computer Systems*, Burlington, Digital Press, 1992.

[22] P. Ch. Kandellakis, and A. A. Shvartsman, *Fault-Tolerant Parallel Computations*, Boston, Kluwer, 1997.

[23] J. R. Heath, P. J. Keukes, G. S. Snider, and R. S. Williams, “A defect-tolerant computer architecture: Opportunities for nanotechnology,” *Science*, vol. 280, Jun. 12, 1998, pp. 1716–1721.

[24] S. Tatapudi, and V. Beiu, “Split-precharge differential noise immune threshold logic gate (SPD-NTL),” in J. Mira, and J. R. Álvarez (Eds.): *Artificial Neural Nets Problem Solving Methods (Proc. IWANN’03*, Menorca, Spain, Jun. 3-6), Springer, LNCS 2687, 2003, pp. 49–56.

[25] A. Schmid, and Y. Leblebici, “Robust circuit and system design methodologies for nanometer-scale devices and single-electron transistors,” *Proc. IEEE-NANO’03*, San Francisco, CA, USA, Aug. 12-14, 2003, vol. 2, pp. 516–519.

[26] A. Schmid, and Y. Leblebici, “A modular approach for reliable nanoelectronics and very-deep submicron circuit design based on analog neural network principles,” *Proc. IEEE-NANO’03*, San Francisco, CA, USA, Aug. 12-14, 2003, vol. 2, pp. 647–650.

[27] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "Fault tolerance techniques for wireless ad hoc sensor networks," *Proc. IEEE Sensors*, 2002, pp. 1491–1496.

[28] A. J. KleinOsowski, and D. J. Lilja, "The NanoBox project: Exploring fabrics of self-correcting logic blocks for high defect rate molecular device technologies," *Proc. IEEE Annual Symp. VLSI ISVLSI'04*, Lafayette, LA, USA, Feb. 19-20, 2004, pp. 19–24.

[29] R. Reischuk, and B. Schmeltz, "Area efficient methods to increase the reliability of combinatorial circuits," *Proc. Intl. Symp. Theoretical Aspects of Comp. Sci. STACS'89*, Paderbon, Germany, Feb. 1989, Springer, LNCS vol. 349, pp. 314–326.

[30] R. Reischuk, and B. Schmeltz, "Area efficient methods to increase the reliability of circuits," in B. Monien, and T. Ottmann (Eds.): *Data Structures and Efficient Algorithms*, Springer, LNCS vol. 594, 1992, pp. 363–389.

[31] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating reliability of defect tolerant architecture for nanotechnology using probabilistic model checking," *Proc. Intl. Conf. VLSI Design VLSID'04*, Mumbai, India, Jan. 5-9, 2004, pp. 907–912.

[32] PRISM publications: http://www.cs.bham.ac.uk/~dxp/prism/publications.html

[33] PRISM case studies: http://www.cs.bham.ac.uk/~dxp/prism/casestudies/nand.html

[34] D. Bhaduri, and S. K. Shukla, "NANOLAB: A tool for evaluating reliability of defect-tolerant nano architectures," *Proc. IEEE Annual Symp. VLSI ISVLSI'04*, Lafayette, LA, USA, Feb. 19-20, 2004, pp. 25–31.

[35] 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, Nov. 9-13, 2003, pp. 480–486.

[36] T. J. Dysart, and P. M. Kogge, "Strategy and prototype tool for doing fault modeling in a nano-technology," *Proc. IEEE-NANO'03*, San Francisco, CA, USA, Aug. 12-14, 2003, vol. 2, pp. 356–359.

[37] S. Roy, V. Beiu, and M. Sulieman, "Reliability analysis of some nano architectures," presented at the *Special Workshop on Neural Inspired Architectures for Nanoelectronics, Neural Information Processing Systems NIPS'03*, Whistler, Canada, Dec. 12-13, 2003. Available: http://www.eecs.wsu.edu/~vbeiu/workshop_nips03/Presentations/S_Roy.pdf

[38] S. Roy, and V. Beiu, "Multiplexing schemes for cost effective fault tolerance," *Proc. IEEE Conf. Nanotech. IEEE-NANO*, Munich, Germany, Aug. 17-19, 2004, in press.

[39] H. L. Hughes, and J. M. Benedetto, "Radiation effects and hardening of MOS technology: Devices and circuits," *IEEE Trans. Nuclear Sci.*, vol. 50, Jun. 2003, pp. 500–521.

[40] M. Lan, A. Tammineedi, and R. Geiger, "A new current mirror layout technique for improved matching characteristics," *Proc. IEEE Midwest Symp. Circ. and Sys. MWSCAS'99*, Aug. 1999, vol. 2, pp. 1126–1129.

[41] M. Lan, and R. Geiger, "Gradient sensitivity reduction in current mirrors with non-rectangular layout structures," *Proc. IEEE Intl. Symp. Circ. and Sys. ISCAS'00*, May 2000, vol. 1, pp. 687–690.

[42] R. L. Dobrushin and S. I. Ortyukov, "Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inform. Transm.,* vol. 13, 1977, pp. 59–65.

[43] R. L. Dobrushin and S. I. Ortyukov, "Upper bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inform. Transm.,* vol. 13, 1977, pp. 203–218.

[44] T. Feder, "Reliable computation by networks in the presence of noise," *IEEE Trans. Inform. Theory*, vol. 35, May 1989, pp. 596–571.

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

[46] R. Reischuk, and B. Schmeltz, "Reliable computation with noisy circuits and decision trees: A general $n\log n$ lower bound," *Proc. Intl. Symp. Foundations of Comp. Sci. FOCS'91*, San Juan, Puerto Rico, Oct. 1-4, 1991, pp. 602–611.

[47] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, "Computing with noisy information," *SIAM J. Comput.*, vol. 23, 1994, pp. 1001–1018.

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

[49] W. S. Evans, "Information Theory and Noisy Computation," Ph.D. dissertation, Univ. of California at Berkeley, *ICSI Tech. Rep. TR-94-57*, Nov. 1994.
Available http://www.cs.ubc.ca/~will/papers/thesis.pdf

[50] A. Gál, "Combinatorial Methods in Boolean Function Complexity," Ph.D. dissertation, Dept. of Comp. Sci., Univ. of Chicago, USA, Aug. 1995.
Available http://www.cs.utexas.edu/users/panni/thesis.ps

[51] W. S. Evans, and N. Pippenger, "On the maximum tolerable noise for reliable computations by formulas," *IEEE Trans. Inform. Theory*, vol. 44, May 1998, pp. 1299–1305.

[52] A. Gál, and A. Rosén, "A theorem on sensitivity and applications to private computations," *SIAM J. Comput.*, vol. 31, 2002, pp. 1424–1437. Preliminary version in

Proc. ACM Symp. Theory of Comput. STOC'99, Atlanta, GA, USA, May 1-10, 1999, pp. 348–357.

[53] W. S. Evans, and L. J. Schulman, "On the maximum tolerable noise of *k*-input gates for reliable computations by formulas," *IEEE Trans. Inform. Theory*, vol. 49, Nov. 2003, pp. 3094–3098.

[54] Y. Qi, J. Gao, and J. A. B. Fortes, "Probabilistic computation: A general framework for fault-tolerant nanoelectronic systems," *Tech. Rep. TR-ACIS-03-002*, ECE Dept., University of Florida, Gainesville, FL, USA, Nov. 28, 2003.
Available: http://www.acis.ufl.edu/techreports/acis03002.pdf

[55] P. Sivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on soft error rate of combinatorial logic," *Proc. Intl. Conf. Dependable Sys. and Networks DSN'02*, Washington, DC, Jun. 23-26, 2002, pp. 389–398.

[56] P. Sivakumar, S. W. Keckler, C. R. Moore, and D. Burger, "Exploiting microarchitectural redundancy for defect tolerance," *Proc. Intl. Conf. Comp. Design ICCD'03*, San Jose, CA, Oct. 13-15, 2003, pp. 481–488.

[57] K. K. Likharev, "Single-electron devices and their applications," *Proc. IEEE*, vol. 87, Apr.1999, pp. 606–632.

[58] M. Sulieman, and V. Beiu, "Design and analysis of SET circuits: Using MATLAB and SIMON," *Proc. IEEE-NANO'04*, Munich, Germany, Aug. 17-19, 2004, in press.

[59] V. Beiu, "A novel highly reliable low-power nano architecture: When von Neumann augments Kolmogorov," *Proc. IEEE Intl. Conf. App.-specific Sys., Arch. and Processors ASAP'04*, Galveston, USA, Sep. 27-29, 2004, pp. 167–177.

[60] M. Goldmann, J. Håstad, and A. Razborov, “Majority gates vs. general weighted threshold gates,” *Proc. Structure in Complexity Theory Conf. CoCo’92*, Boston, MA, USA, Jun. 22-25, 1992, pp. 2–13.

[61] M. Goldmann, and M. Karpinski, “Simulating threshold circuits by majority circuits,” *Proc. ACM Symp. Theory of Computing STOC’93*, San Diego, CA, USA, May 16-18, 1993, pp. 551–560.

Sandip Roy (S’01–M’04) received his B.S. from the University of Illinois, Urbana-Champaign, in 1998 and his S.M. and Ph.D. at the Massachusetts Institute of Technology in 2000 and 2003, respectively, all in electrical engineering. He is currently Assistant Professor of Electrical Engineering at the Washington State University (WSU). He has held numerous internship positions, including at the National Aeronautics and Space Administration and at the National Institute of Standards and Technology. He is author of more than 20 refereed publications, and has given several invited talks. His research interests are in the areas of systems and control, with a particular focus on the modeling and design of complex networks. He is currently faculty mentor for the WSU IEEE Student Branch.

Valeriu Beiu (S’92–M’95–SM’96) received the M.Sc. in computer engineering from the “Politehnica” University of Bucharest (Romania), in 1980, and the Ph.D. with *summa cum laude* in electrical engineering from the Katholieke Universiteit Leuven (Belgium), in 1994.

After graduating, he has worked for two years on high-speed CPUs and FPUs at the Research Institute for Computer Techniques (Bucharest), before returning to the “Politehnica” University of Bucharest. He has received five fellowships: Fulbright (1991), Human Capital and Mobility (1994-1996) with King’s College London (project title: *“Programmable Neural Arrays”*), Director’s Funded Postdoc (1996–1998) with Los Alamos National Laboratory (project title: *“Field Programmable Neural Arrays,”* under the Deployable Adaptive Processing Systems initiative), and Fellow of Rose Research (1999–2001). He was a founder and CTO of RN2R (1998–2001), and since 2001 he is an Associate Professor with the School of EE&CS,

Washington State University (Pullman, USA). He was the PI of 34 research contracts, holds 11 patents, received 32 grants, gave over 100 invited talks, and authored over 120 papers in refereed journals and international conferences. He is the author of a chapter on digital integrated circuit implementations (of neural networks) in the *Handbook of Neural Computation*, and of a forthcoming book on the VLSI complexity of discrete neural networks. His main research interests are: VLSI design of high–performance innovative computer architectures and algorithms (massively parallel, adaptive/reconfigurable), their optimized designs—inspired by systolic arrays, neural networks, quantum, or a mixture of them—and their application for computational intensive tasks. I like to take abstract concepts for very difficult but practical applications, turn them into efficient algorithms, and then design innovative VLSI circuits performing them optimally (*i.e.*, at ultra-high speeds, with very low-power, with high reliability, etc.). I am very interested by emerging nanoelectronics and in particular by nano architectures (massively parallel, adaptive / reconfigurable, regular, fault-tolerant, neural inspired), by their optimized designs—inspired by cellular and systolic arrays, artificial neural networks, or combinations of these—and by their application to computational intensive tasks.

Dr. Beiu is a member of INNS, ENNS, ACM, and MCFA, has organized 11 conferences and 22 conference sessions, has received 5 Best Paper Awards, and has been the Program Chairman of the IEEE Los Alamos Section (1997).

FIGURE CAPTIONS

Fig. 1. NAND-2 von Neumann multiplexing.

Fig. 2. Comparison of several techniques showing the allowable failure rate per device p_f with respect to the redundancy factor R (adapted from [3] by highlighting the ranges of very small and small redundancy factors).

Fig. 3. The variation of the error thresholds of NAND and MAJ gates with respect to their fan-in.

Fig. 4. (a) Generic MAJ-3 MUX stage. (b) An example for $N = 5$.

Fig. 5. Maximum allowed failure probability of MAJ-3 MUX at small redundancy factors ($R < 100$), for $N_c = 10^6$.

Fig. 6. Maximum allowed failure probability MAJ-3 MUX(N, k) at small redundancy factors ($R < 100$), for $N_c = 10^6$.

Fig. 7. Comparison of MAJ-3 MUX and enhanced MAJ-3 MUX(N, k). (a) At small redundancy factors ($R < 100$). (b) At very small redundancy factors ($R < 10$).

Fig. 8. (a) The MAJ-3 MUX($3, \infty$) structure used for testing. (b) Probability of failure versus variations for one MAJ-3 SET gate, and for the MAJ-3 MUX($3, \infty$) with high-matching MAJ-3 SET gates.

Fig. 9. A capacitive SET MAJ-3 MUX design was compared with a high-matching bias capacitors version (generated with MATLAB and simulated in SIMON, a Monte Carlo simulator).

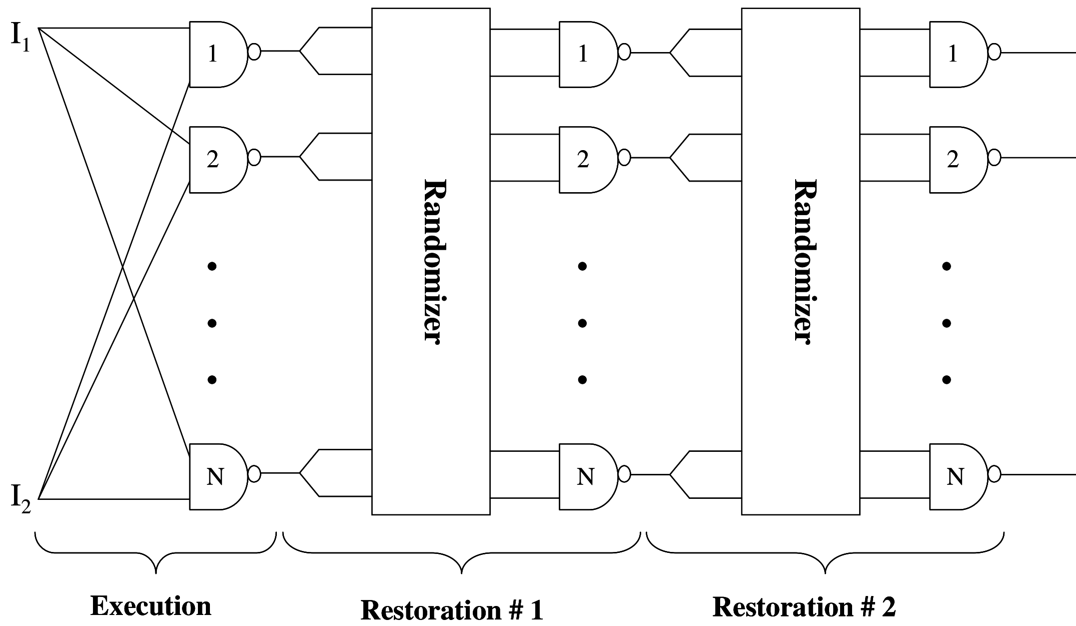


Fig. 1. NAND-2 von Neumann multiplexing.

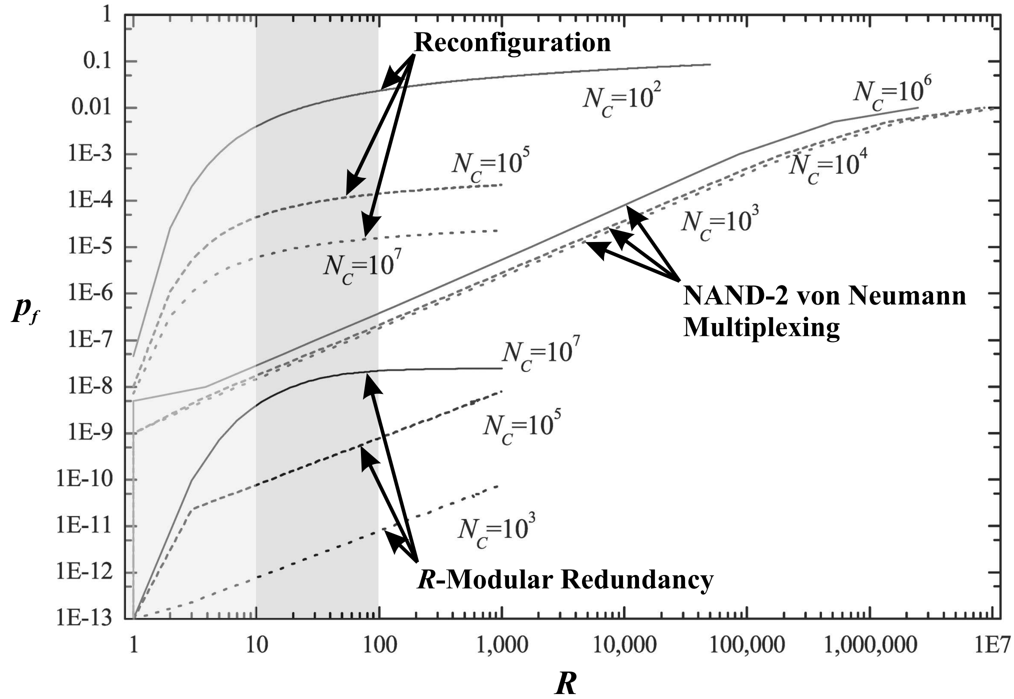


Fig. 2. Comparison of several techniques showing the allowable failure rate per device p_f with respect to the redundancy factor R (adapted from [3] by highlighting the ranges of very small and small redundancy factors).

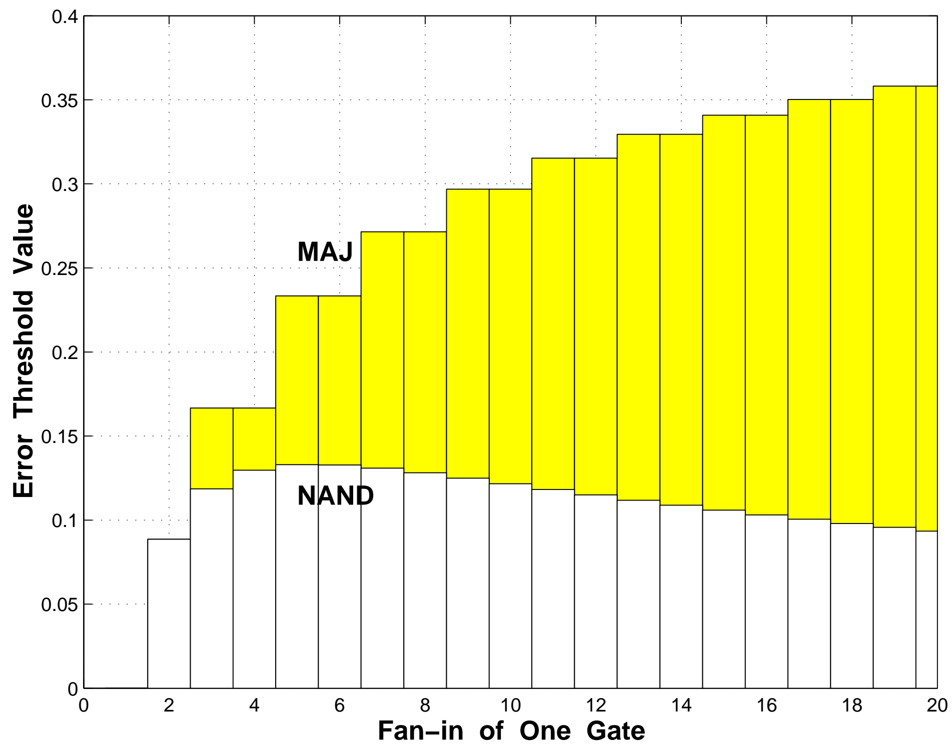
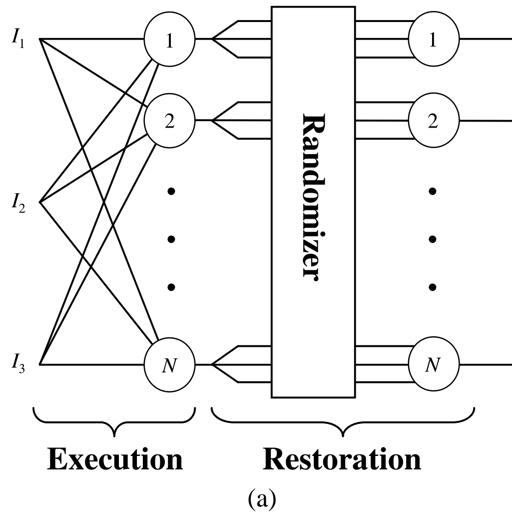
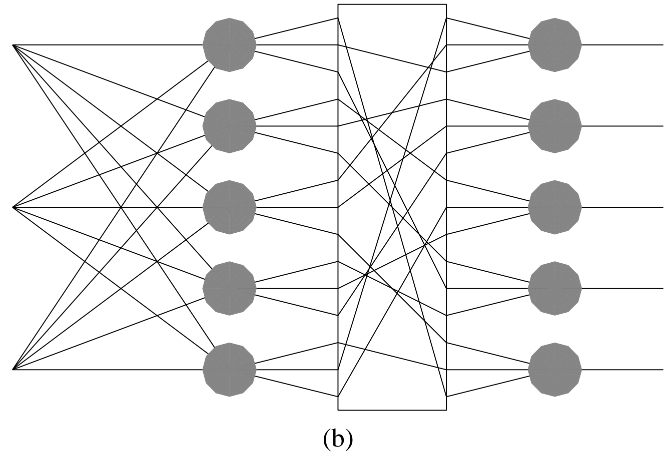


Fig. 3. The variation of the error thresholds of NAND and MAJ gates with respect to their fan-in.



(a)


(b)

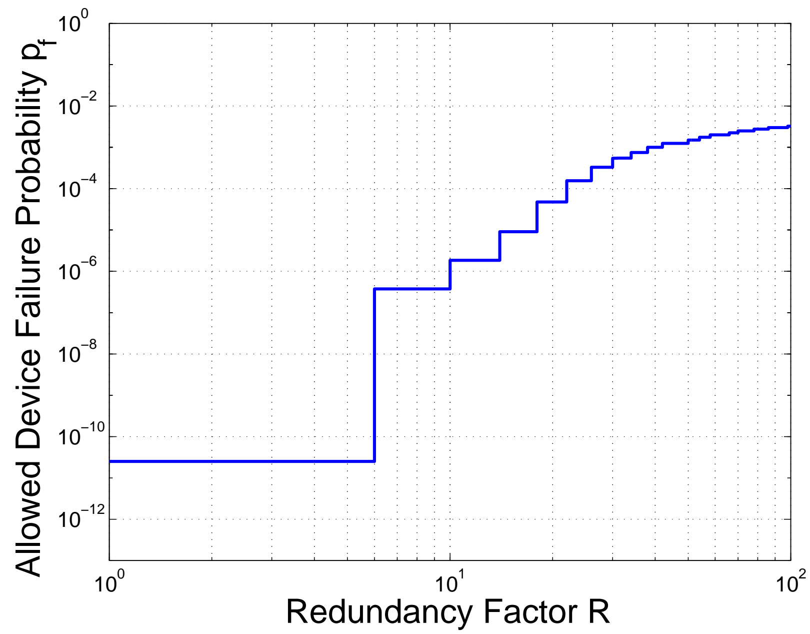


Fig. 5. Maximum allowed failure probability of MAJ-3 MUX at small redundancy factors ($R < 100$), for $N_c = 10^6$.

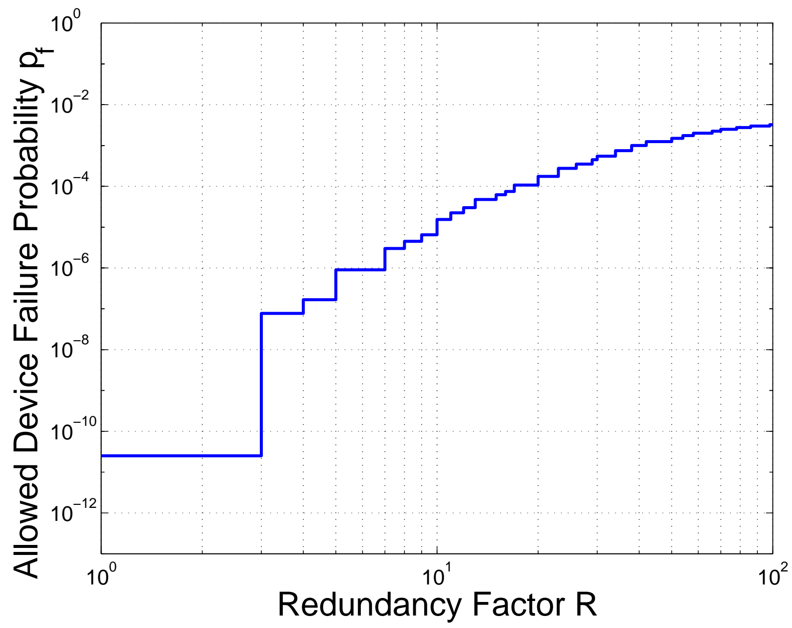
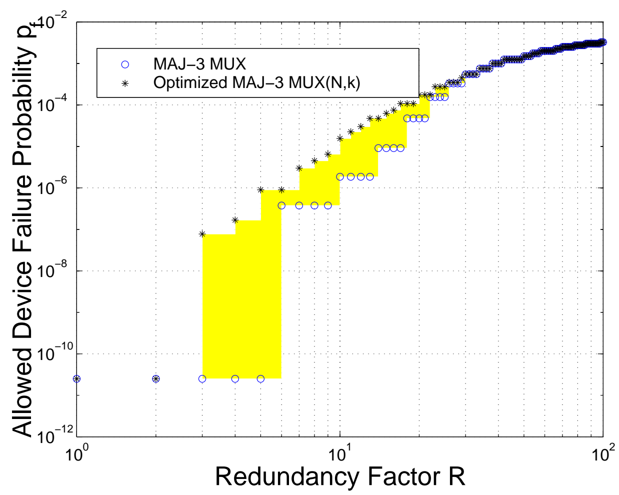
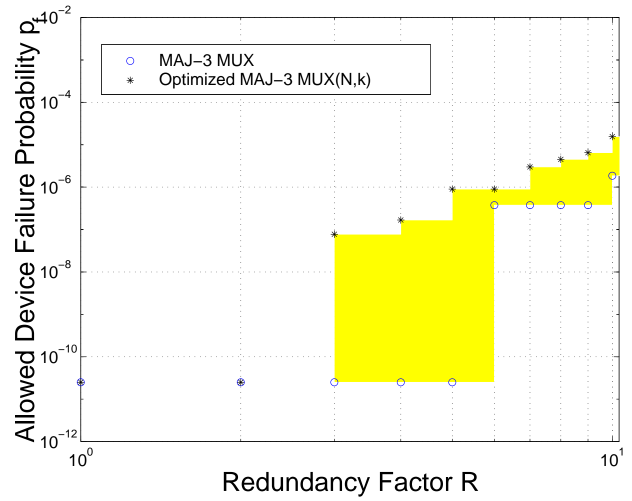


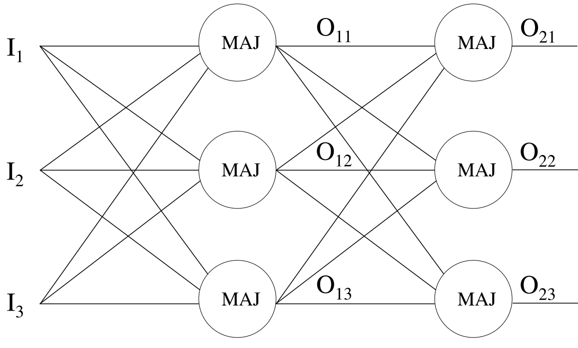
Fig. 6. Maximum allowed failure probability MAJ-3 MUX(N, k) at small redundancy factors ($R < 100$), for $N_c = 10^6$.



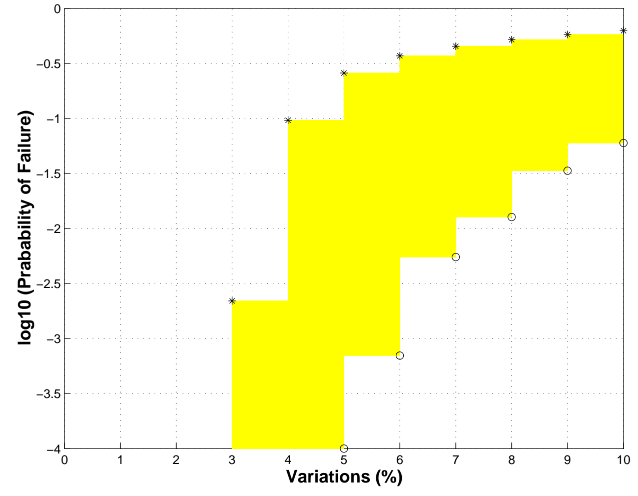
(a)



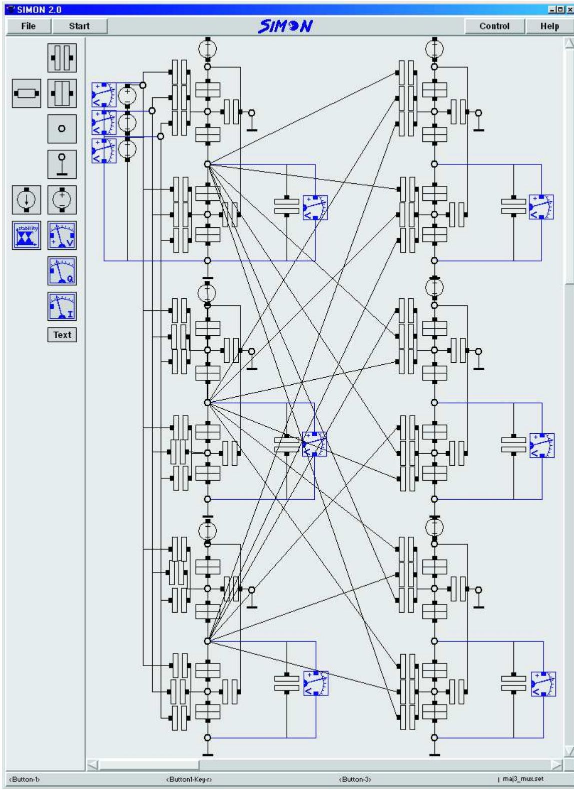
(b)



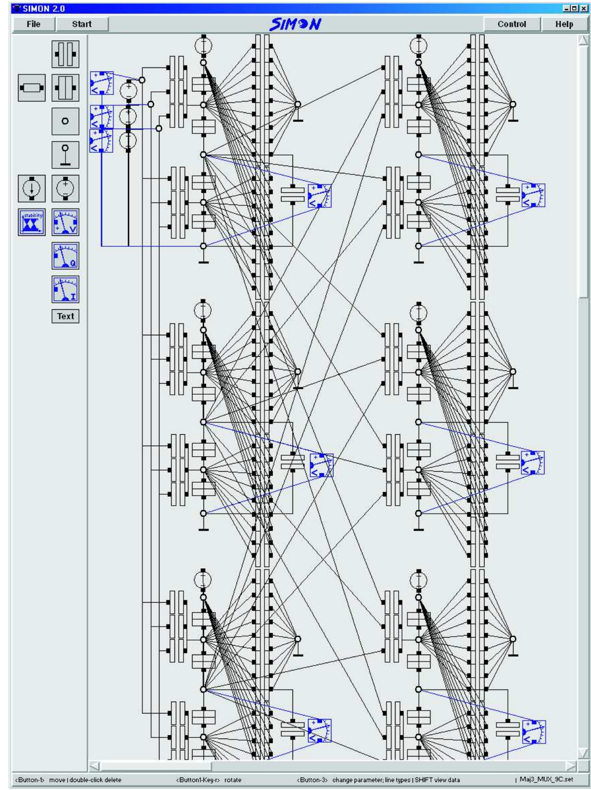
(a)



(b)



(a)



(b)

TABLE I

MAJ-3 MUX AND OPTIMIZED MAJ-3 MUX(N, κ) FOR VERY SMALL REDUNDANCY FACTORS $R = 3 \ldots 10$

Redundancy Factor	Maximum Allowed Device Failure Probability		Improvement Factor	(N, k) Optimized Pair
	MAJ-3 MUX	Optimized MAJ-3 MUX(N, k)		
3	2.5×10^{-11}	7.8×10^{-8}	**3100**	(3, ∞)
4	2.5×10^{-11}	1.7×10^{-7}	**6700**	(3, 3)
5	2.5×10^{-11}	9.0×10^{-7}	**36,000**	(5, ∞)
6	3.8×10^{-7}	9.0×10^{-7}	2.4	(5, ∞)
7	3.8×10^{-7}	3.0×10^{-6}	8	(7, ∞)
8	3.8×10^{-7}	4.5×10^{-6}	12	(7, 10)
9	3.8×10^{-7}	6.5×10^{-6}	17.3	(7, 4)
10	1.9×10^{-6}	1.6×10^{-5}	8.4	(9, 10)

Sandip Roy

Valeriu Beiu