The Vanishing Majority Gate
Trading Power and Speed for Reliability

Valeriu Beiu[1], Snorre Aunet[2], Ray Robert Rydberg III[1], Asbjørn Djupdal[3], and Jabulani Nyathi[1]

[1] School of EE&CS, Washington State University, Pullman, Washington, USA
[2] Department of Informatics, University of Oslo, Norway
[3] Department of CS&IT, Norwegian University of Science and Technology, Norway

***Abstract*—In this paper we are going to explore low-level implementation issues for fault-tolerant adders based on multiplexing using majority gates (MAJ).**

We shall analyze the particular case of a 32-bit ripple carry adder (RCA), as well as different redundant designs using MAJ-3 (MAJ of fan-in 3) multiplexed RCAs: (i) with classical MAJ-3 gates in the restorative stages; (ii) with inverters driven by short-circuited outputs at each restorative stage; and finally, (iii) only with short-circuited outputs at each restorative stage. From one solution to the next, the restorative MAJ-3 gates get simpler and simpler. These simplifications translate into different speeds and power consumptions; challenging aspects of future nanoelectronics. All these circuits have been designed and simulated in subthreshold. The speed and power will be reported and compared for designs in 0.18 µm as well as in 70 nm (using the Berkeley Predictive Technology Model). The results reveal interesting power-speed-reliability tradeoffs.

In two of these designs, depending on the way the MAJ-3 function is implemented, defects translate into increased power, and suggest a (simple) way of detecting them. A detection circuit can trigger reconfiguration at a higher level, leading to a seamless transition from a fault/defect-tolerant circuit to a defect-tolerant system. The main advantage of such an approach would be that the reconfiguration could be done on-line, i.e., while the circuit is still operating correctly.

***Index Terms*—Architecture, defect- and fault-tolerance, majority logic, multiplexing, power.**

I. Introduction

Scaling of CMOS into the nanometer range raises many challenges [1], [2]. The development of novel nanodevices brings promise for improvements in performance, yet it also leads to new challenges, including both the increasing power consumption, and the need for architectures that reduce the uncertainty inherent to (nano)computations [3]–[5]. That is why, fault- and defect-tolerant architectures have recently received revived attention in the nanotechnology community [6]–[10]. One well-known approach for developing fault-tolerant architectures in the face of uncertainties (both *defects* and transient *faults*) is to incorporate spatial and/or temporal redundancy. Among the redundant design schemes, we should mention here: modular redundancy, cascaded modular redundancy, multiplexing (MUX, including von Neumann multiplexing [11] and parallel restitution PAR-REST [10]), as well as reconfigurability [6], [9], [12].

Reliable operation of a circuit can be achieved using redundancy at many different levels: at the device level [13], [14]; at the gate level [15], [16]; at the block level [17]; in time; and in communication (through encoding, e.g., [18]) (see also [2] and [4]–[10]). We note here that all of these have in common that improved reliability is traded off for increased chip *area* and higher *connectivity*. These lead to higher *power consumptions*, and can also slow down the computations.

The most common way of quantify redundancy is to use a *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. Cost-effectiveness constraints dictate that redundancy factors must be small, or better very small. Still, the increase in circuit *area* rather than increase in *size* is a more significant measure of redundancy, as suggested in [19], [20] (where the authors also show how encoding in combination with replication can be used to minimize circuit area).

In this paper the focus will be on the gate and block levels. Section II provides a review of multiplexing (MUX) schemes starting from the early work of von Neumann and detailing recent variations and enhancements. A comparison of MUX with other techniques using redundancy shows significant advantages for MUX. The use of MAJ gates in MUX improves over MUX schemes based on NAND gates. That is why, in Section III we analyze different implementations for MAJ functions. Beside reliability, power is becoming an important issue, hence we are going to discuss MAJ-3 gates targeted for subthreshold operation. Section IV will put all of these together, and present different configurations of a MAJ-3 MUX 32-bit adder. Only serial addition will be considered, as it outperforms parallel addition when operated in subthreshold. Simulation results will be presented and discussed before concluding.

II. On Multiplexing

In [11], von Neumann introduced the multiplexing redundancy algorithm MUX as a plausible representation for reliable (neural-inspired) computation. The MUX algorithm aims to improve the reliability of a sequence of computations. This 'multiplexing' of each computation serves to contain error propagation, by selecting the more-likely result at each stage. MUX was developed for arbitrary gates, including MAJ and NAND gates (see Fig. 1 showing the executive stage followed by two restorative stages of a NAND-2 MUX). However, a detailed reliability analysis was performed for two-input NAND (NAND-2) gates only, assuming independent gate failures and very large redundancy factors. The performance of NAND-2 MUX was compared with the performance of other fault tolerance techniques in [4]–[8]. In [8], NAND-2 MUX was analyzed at small to moderate redundancy factors of 30, 300, and 3000. NAND-2 MUX has been analyzed using a CAD tool in [21]. The results reported in [21] show that for small redundancy factors the theoretical results from [8] are inaccurate.

The PAR-REST scheme [10] is of particular interest. The authors distinguish PAR-REST from NAND-2 MUX based on the fact that the computations are not collapsed after each layer of the circuit (see [10] for details) and that *restorative stages* are only used periodically. They show that PAR-REST can significantly improve upon NAND-2 MUX for small to moderate R. A similar approach was taken in [22], [23], where MAJ-3 gates were used instead on NAND-2.

The issue of which gate to use is debatable. MUX can be applied to any logic gate, but for each new gate and even for another fan-in value, the analysis must be redone. Following is a list of pros and cons.

- NAND MUX requires two restoration stages, while MAJ requires only one. This leads to less area, shorter delay, less power, and less energy. Still, this is not as clear as it seems, by proper design one NAND restoration could be eliminated.
- MAJ has an error threshold higher NAND (see Fig. 2). In this figure, the error threshold for MAJ-k gates (for k odd) is the one determined in [24], [25], while the error threshold for NAND-k gates was recently proven in [26]. The figure also suggests that MAJ gates of large fan-ins are (theoretically) better for improving reliability (see also [27]).
- Finally, MAJ-3 MUX can achieve accurate computations for gate failure probabilities $q_{\text{MAJ-3}} < 0.0197$ (see [23]). This outperforms the NAND-2 gate failure probabilities $q_{\text{NAND-2}} < 0.0107$ (see [10] for a relevant discussion).

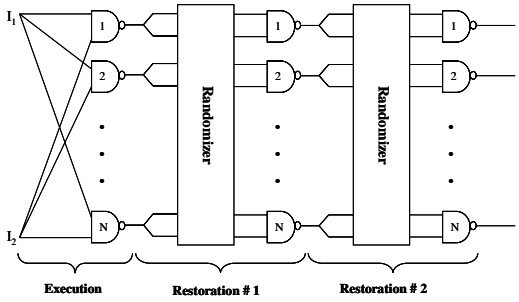

Fig. 1. NAND-2 von Neumann multiplexing.

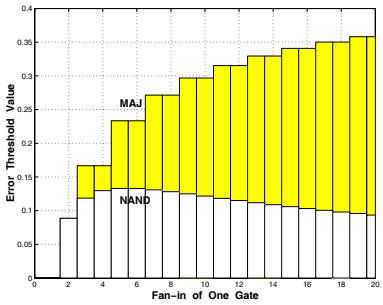

Fig. 2. The error thresholds for NAND and MAJ gates with respect to their fan-in.

The idea of using MAJ was presented in the original article of von Neumann [11]. Still, exact evaluation of the probability of failure at very small redundancy factors was analyzed and proven only recently [22], [23].

A single MAJ-3 MUX logic computation is presented in Fig. 3(a). The MUX computation comprises an *executive stage* and a *restorative stage*. The executive stage repeats the desired logic 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. 3(b)) the outputs from the executive stage, and then chooses the majority of each randomly-chosen set of three signals using a set of N MAJ-3 gates, to generate the N final outputs. This restoration is central to the global performance of the MUX scheme. The purpose of the restorative stage in MUX is to reduce error propagation from a logic computation's input to its output, by selecting the more

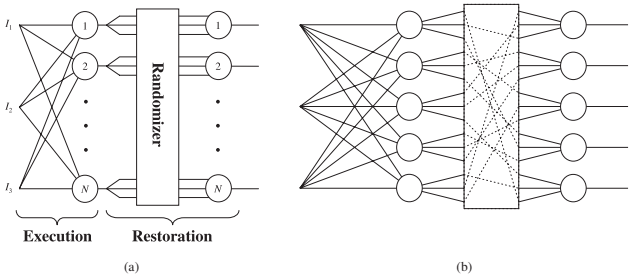

Fig. 3. (a) Generic MAJ-3 MUX stage, and (b) the $N = 5$ case.

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.

If one is seeking the best-performing architecture for a particular redundancy factor R, it has been shown that the standard MUX algorithm can be improved on by applying the restorative stage on only some computations, while simultaneously increasing the bundle size N. This idea was used in [23] to improve MAJ-3 MUX, but the same principle can be applied when using any other type of gate or combinatorial logic block. Let us 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). The enhanced MAJ-3 MUX(N, k) architecture is one in which an executive stage with bundle size N is used for all computations, and a restorative stage is applied only on every k^{th} stage (i.e., for computations with logical depth k, $2k$, … — while in general the restorative stages could be distributed unevenly). The redundancy factor introduced by a MAJ-3 MUX(N, k) architecture is $R = N + N / k$. By placing the restorative stage only every k^{th} stage, the bundle size N can be increased to $N^{+} = [2k/(k+1)]\times N$ for the same redundancy factor R. Obviously, this not only maximizes reliability, but also reduces delay, area, and power. A comparison of R-modular redundancy (RMR), NAND-2 MUX, reconfiguration, and the enhanced MAJ-3 MUX can be seen in Fig. 4.

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 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 (2.3×10^{-4} versus about 1.5×10^{-4}), while achieving a factor of 4.25 at $R = 100$ (1.7×10^{-3} versus about 4×10^{-4}).

These fresh results and enhancements on MUX show that the technique is able to start competing with reconfiguration (which is not able to deal with faults) for quite small redundancy factors, if the reconfiguration is performed on large blocks ($N > 10^3$).

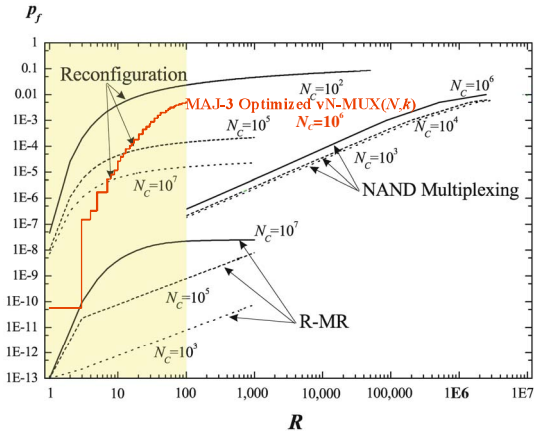

Fig. 4. Comparison of RMR, NAND-2 vN-MUX, reconfiguration, and the enhanced MAJ-3 MUX(N, k).

Novel redundancy techniques that combine device-level ([13], [14]) and gate-level design ideas have also been presented [15]. In [16], 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 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. This solution presented in [15] precedes [16], and also has the advantage of lower power consumption for the case of fault-free operation. Other low-level approaches which we should mention here belong to the larger class of rad-hard by design [28], and high matching techniques used in analog circuits [29], [30] (recently used for enhancing the reliability of CMOS TLGs [13], and capacitive SET [14]).

Very recently [18], 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.

III. Majority Gates

Based on the discussion of the previous section, MAJ MUX schemes seem to have an edge over NAND MUX ones. It then becomes a question of how we implement the MAJ function. Before going further, we mention that MAJ functions can always be replaced by minority functions if the inputs are inverted and vice versa. That is why we are going to refer to these implementations as MAJ gates, even if sometimes the function they implement is the minority function.

MAJ gates can be implemented in many different ways. A standard CMOS implementation is the well-known "mirrored adder" [31] (see Fig. 5). Domino logic gates could be used to improve the speed, but raise problems with distributing the clock, higher power consumption, and reduced noise margins (sensitivity to variations and clock skew). Differential logic could be another alternative, but with scaling the leakage currents are going to be higher (as compared with the other logic styles) [32]. Using pass transistors/gates as a multiplexor followed by an inverter (as buffer) is a very simple solution. This type of gates has been recently shown to work reliably even in subthreshold [33]. They are low power, but also kind of slow. Pseudo-nMOS (or its variations) have long been known and used for implementing threshold logic gates [34]. These can be very fast, but power hungry, and the noise margins are small. These gates are also sensitive to variations.

MAJ-3 gates operating in subthreshold are the basic building blocks to be used in this paper. This is because subthreshold operation is considered to consume less power than any other known low-power solution, even lower than energy recovery logic [35]. Because reduction of power consumption is mandatory for future scaled CMOS [36], subthreshold operation is very likely to play an important role in the design of circuits on the scaling path towards the 10 nm node [37]. That is why in this paper we suggest using MAJ-3 in subthreshold [38]–[53]. Combining such gates with low-level redundancy (improved matching and fault-tolerance) was suggested in [15], by short-circuiting the outputs of three gates, so no voter (MAJ-3) was required.

In Figure 5 the second gate from left [38] is a floating gate structure depending on a somewhat exotic UV-postprocessing technique (probably not suitable for future scaled CMOS). This is due to the dependence on nonvolatile analog memory from charges depleted on the floating gates, through UV activated conductances [43].

The gate from [41], [42], is the third gate in Fig. 5. It exploits the transistor as a four terminal device, using the wells to control the threshold for changing the functionality in real time, and/or some automatic body biasing [44]. This is able to also implement NOR-3 and NAND-3 [42]. The larger relative transconductance in subthreshold, compared to the classical above threshold region, makes this possible. This is not the case for other circuits based on inverters with short-circuited outputs (e.g., [45]), as they are not intended for subthreshold, and use the transistors as 3 terminal devices.

IV. Fault-Tolerant Adders

The particular example we are going to use in this paper is a 32-bit adder. Many different alternative designs are possible, starting from the serial ripple carry adder (RCA) and going towards parallel implementations [46]–[50]. It is commonly accepted that the slowest one is the RCA, while Kogge-Stone [47] (KS) is expected to be the fastest. Classical CMOS gates are almost never used when fast addition is in the picture, and domino gates are the norm, with threshold logic gates advocated for even higher speeds [51], or for optimal mixed combinations with domino logic [52].

An RCA and a KS have very recently been analyzed [53] when operating in subthreshold (at 100 nm and 70 nm). The main conclusions are that:

- the wires are reducing the speed advantage of the KS over RCA from 4.5x to 2.2x;
- the speed of the KS at a given V_{DD} can be matched by RCA at a slightly larger V_{DD} (10% to 20%);
- at equal speeds, the RCA still maintains a clear power and energy advantage [53].

The integration of MAJ-3 MUX with an adder was discussed in [54]. A KS adder can be seen in Fig 6(a), and a MAJ-3 MUX(3,3) enhanced KS adder can be seen in Fig. 6(b). The connectivity pattern gets complex, and the longer wires will contribute both to increasing the delay and the switching power. Based on the above factors, and on the simulation results from [53], we decided to focus on RCA. It is also much easier to integrate MAJ-3 MUX with an RCA (see Fig. 7, and compare with Fig. 6(b)).

The main block of an RCA is the well-known full adder (FA). Many investigations for optimizing the FA at the gate level have been reported [55]. The results for the many FAs investigated are not directly translatable for subthreshold operation. We have investigated an FA based

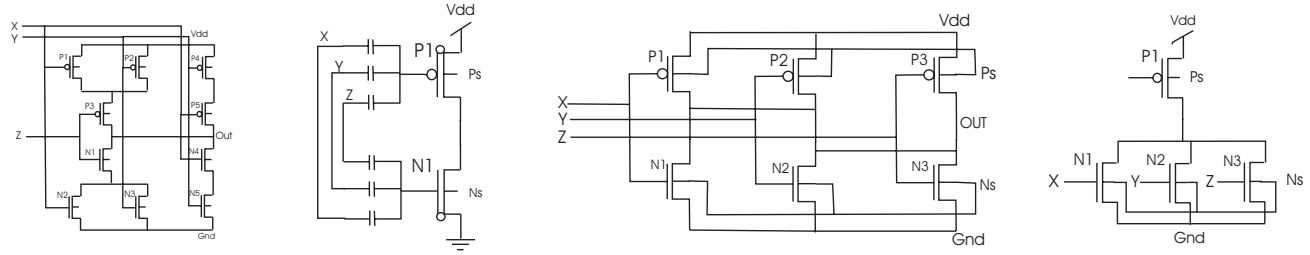

Fig. 5. Different MAJ-3 gates from left to the right: [31], [38], [42], [34].

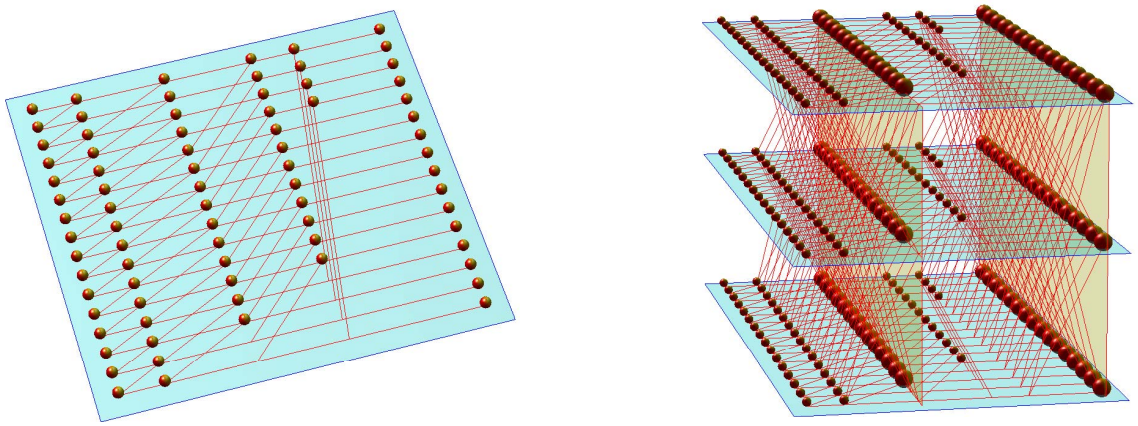

Fig. 6. (a) Kogge-Stone adder; and (b) MAJ-3 MUX(3,3) implementation of the Kogge-Stone adder [54].

on MAJ-3 gates implemented as "output-wired inverters" [53]. We have also experimented with combinations of gates for optimizing the FA. A very low power FA in subthreshold uses a "mirrored adder" for the MAJ-3 (computing the carry-out), and two pass-gates (like the ones in [33]) for implementing the XOR-3 (computing the sum). This is the FA that we have used in all the RCAs in this paper (see Fig. 7). The XOR-3 is very low power and kind of slow, but it is not in the critical path of the RCA. A standard CMOS implementation of an XOR-3 not only dissipates more than the pass-gate solution (in subthreshold), but is also more sensitive to variations and skewed inputs.

The top drawing in Fig. 7 presents a block diagram of the standard RCA. The MAJ-3 MUX RCA configurations have three parallel FAs per stage, and can be summarized as follows:

- use three RCAs in parallel;
- use three MAJ-3 gates to 'vote' on the carry-out coming from the three FAs at position *i*;
- use the output of each of these three MAJ-3 to drive the three carry-in of the three FAs at bit position *i*+1.

These are mapped into the next three structures in Fig. 7, with each subsequent structure being less complex than the previous.

- The first of the three structures properly uses MAJ-3 gates for the restorative stages (represented as circles). This will double the delay and increase power.
- The complexity of the second structure is reduced with the outputs of the FAs being tied together and fed as inputs to the restorative inverters. The MAJ-3 gate is now reduced to several wires and an inverter. This solution is faster, and will dissipate less than the pervious one, as long as there are no faults/defects. If there are faults/defects, the fighting on the carry-out will increase the power consumption of the inverters which will try to restore the logic levels.
- The simplest structure of all eliminates the restorative inverters and uses the next stage of the FAs to provide signal restoration. The MAJ-3 has now vanished. Note how the outputs of each structure are shorted. This solution will be the fastest as long as no faults/defects.

Shorting the outputs could result in a path from V_{DD} to ground, resulting in increased current, while placing the gates in parallel and making them drive subsequent stages sequentially increases the signal propagation delay. These three solutions show very different power-delay tradoffs both when working correctly and when faulty. We have tested the structures for stuck-at-faults. This corresponds to a worst case scenario, as in practice a fault/defect would manifest itself as an analog value in between V_{DD} and GND (see [16], [56]). If the number of stuck inputs per

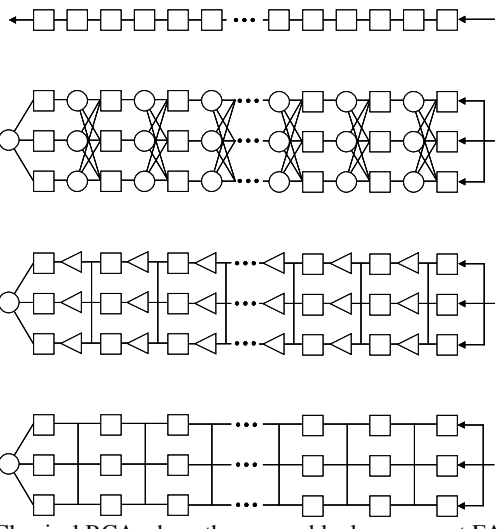

Fig. 7. Classical RCA where the square blocks represent FAs. The three different MAJ-3 MUX RCAs: (i) using MAJ-3 gates (circles) in between FAs; (ii) short-circuiting the outputs of three FAs and using three inverters (triangles) to recover the voltage; and (iii) short-circuiting the outputs of three FAs (the voltage is recovered by the next three FAs).

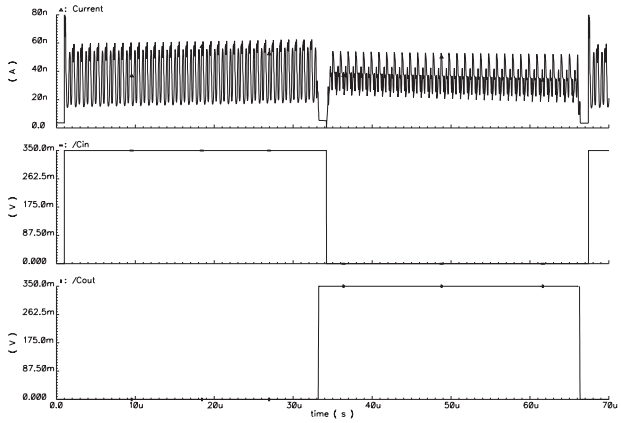

Fig. 8. Current (worst case) for MAJ-3 MUX RCAs using MAJ-3 gates (in 180nm).

stage exceeds one, the condition constitutes immediate failure.

The behavior of the three different MAJ-3 MUX RCAs can be seen in Figs. 8–10, while numerical values are reported in Table I. The traces of Fig. 8 show a worst case with 32 faults (one in every stage), but the current does not increase. The short circuited MAJ-3 MUX RCAs with faults show the current increasing linearly in steps as the number of fights in different stages increase. These traces can be seen in Figs. 9 and 10 for the 180nm node, and in Fig. 11 for the 70nm node (Fig. 11 is equivalent to Fig. 9). It can be seen that a designer has quite a large number of options for trading off power and speed when using MUX.

The results from Figs. 9–11 show a significant current increase when faults occur (see step values in Table I). A current-aware circuit can trigger a reconfiguration process at a higher level. The current work is however aimed at showing that these faults cause significant current changes we can depend on for detecting them. The simulation results are promising as showing that scaling from the 180nm node to the 70nm node results in significantly shorter delays (40x), with small increase of currents (6x), both with and without faults.

V. Conclusions

The paper has analyzed multiplexed adder designs working in subthreshold. The subthreshold operation was employed to address the power challenge. Still, while

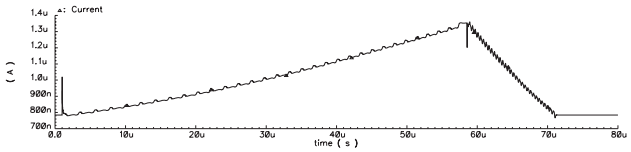

Fig. 10. Current (worst case) for MAJ-3 MUX RCAs with short-circuited outputs (in 180nm).

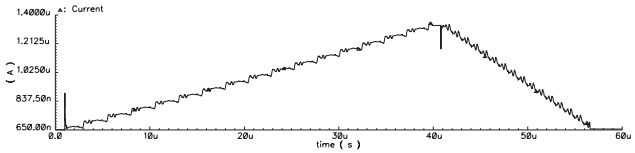

Fig. 9. Current (worst case) for MAJ-3 MUX RCAs using inverters and short-circuiting the outputs (in 180nm).

reducing the voltage supply into the subthreshold region might save the day for power consumption, it will adversely affect reliability. That is why, we proposed and investigated a MAJ-3 MUX architectural approach for 32-bit adders. Serial solutions seem to have an advantage over parallel ones, and also integrate well with MUX. The implementation of the MAJ-3 gates can be done in several different ways, with three being detailed in this paper, namely: classical CMOS gate, short-circuits followed by inverters, or only short-circuits. Simulations have shown that the two solutions relying on short-circuiting the outputs are faster in case of no faults/defects. The delay increases when faults/defects start appearing, but the circuit is still able to function correctly, showing a gradual degradation of its speed performance. The faults/defects also significantly increase current (power). This might be seen as a disadvantage, but can be used as a way of detecting the faults/defects. A current-aware circuit can trigger reconfiguration at a higher level if currents get above a certain threshold. Once the reconfiguration has been achieved, the defective circuit/block can be swapped with a non-defective one, and then shut down.

References

[1] *International Technology Roadmap for Semiconductors*, ITRS, 2004. Available: http://public.itrs.net/

[2] R. Compañó, L. Molenkamp, and D. J. Paul (Eds.), *Technology Roadmap for Nanoelectronics*, EC-FET, 2000. Available: http://www.cordis.lu/esprit/src/melna-rm.htm

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

[4] 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. 2002, pp. 389–

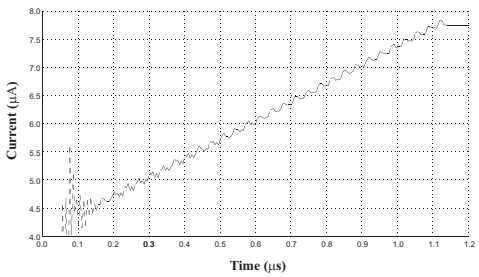

Fig. 11. Current (worst case) for MAJ-3 MUX RCAs using inverters and short-circuiting the outputs (in 70nm).

TABLE I
PERFORMANCES OF DIFFERENT MAJ-3 MUX (REDUNDANT) RIPPLE CARRY ADDERS

Circuit and Technology Node	Delay (us)		Current (nA)	Power (nW)	PDP (fJ)	PDP per stage (fJ)
0.18µm CMOS						
One RCA	17.140		11.68	4.09	69.90	2.18
3 RCAs short-wired	17.230		36.15	12.65	218.00	6.81
3 RCAs short-wired (one RCA stuck-at)	56.550	Max	1353.00	473.60	26779.00	836.90
		Min	784.00			
		Step	17.78			
3 RCAs with inverters	21.260		35.07	12.27	261.00	8.35
3 RCAs with inverters (one RCA stuck-at)	38.790	Max	1337.00	467.950	18152.00	581.00
		Min	655.00			
		Step	21.31			
3 RCAs with MAJ-3	32.150		38.39	13.44	432.00	13.50
3 RCAs with MAJ-3 (one RCA stuck-at)	32.200		34.93	12.23	393.70	12.30
70nm (BPTM)						
One RCA in 70nm	0.543		362.00	72.40	39.31	1.23
3 RCAs with inverters in 70nm	0.652		1053.00	210.60	137.31	4.29
3 RCAs with inverters in 70nm (one RCA stuck-at)	1.074	Max	7751.00	1550.20	1664.91	52.03
		Min	3737.00			
		Step	125.44			

398.

[5] 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. 2003, pp. 481–488.

[6] M. Forshaw, K. Nikolić, and A. S. Sadek, "ANSWERS: Autonomous Nanoelectronic Systems With Extended Replication and Signaling," *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

[7] K. Nikolić, A. S. Sadek, and M. Forshaw, "Fault-tolerant techniques for nanocomputers," *Nanotechnology*, vol. 13, Jun. 2002, pp. 357–362.

[8] J. Han, and P. Jonker, "A system architecture solution for unreliable nanoelectronic devices," *IEEE Trans. Nanotech.*, vol. 1, Dec. 2002, pp. 201–208.

[9] J. Han, and P. Jonker, "A defect- and fault-tolerant architecture for nanocomputers," *Nanotechnology*, vol. 14, Feb. 2003, pp. 224–230.

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

[11] 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.

[12] 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.

[13] 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), Springer, LNCS 2687, Jun. 2003, pp. 49–56.

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

[15] S. Aunet, and M. Hartmann "Real-time reconfigurable threshold elements and some applications to neural hardware," *Proc. Intl. Conf. Evolvable Sys. ICES'03*, Trondheim, Norway, Springer LNCS 2606, Mar. 2003, pp. 365–376.

[16] 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. 2003, vol. 2, pp. 516–519.

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

[18] 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. 2004, pp. 19–24.

[19] 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.

[20] 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.

[21] 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. 2004, pp. 907–912.

[22] 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. 2003. Available: http://www.eecs.wsu.edu/~vbeiu/workshop_nips03/Presentations/S_Roy.pdf

[23] S. Roy, and V. Beiu, "Multiplexing schemes for cost effective fault tolerance," *Proc. IEEE-NANO'04*, Munich, Germany, Aug. 2004. Extended version to appear as "Majority multiplexing: Economical redundant fault-tolerant designs for nano architectures," *IEEE Trans. Nanotech.*, 2005

[24] 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

[25] 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.

[26] 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

[27] R. Reischuk, "Can large fanin circuits perform reliable computations in the presence of faults?," *Theoretical Comp. Sci.*, vol 240, Jun. 2000, pp. 319–335.

[28] 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.

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

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

[31] D. Hampel, K. J. Prost, and N. R. Scheinberg, "Threshold logic using complementary MOS device," U.S. Patent 3 900 742, Jun. 24, 1974.

[32] J. Nyathi, V. Beiu, S. Tatapudi, and D. Betwoski, "A charge recycling differential noise-immune perceptron," *Proc. Intl. Joint Conf. Neural Networks IJCNN'04*, Budapest, Hungary, Jul. 2004, pp. 1995–2000.

[33] B. H. Calhoun, A. Wand, and A. Chandrakasan, "Device sizing for minimum energy operation in subthreshold circuits," *Proc. Custom IC Conf. CICC'04*, Oct. 2004, pp. 95–98.

[34] V. Beiu, J. M. Quintana, and M. J. Avedillo, "VLSI implementation of threshold logic: A comprehensive survey," *IEEE Trans. Neural Networks*, vol. 14, Sep. 2003, pp. 1217–1243.

[35] J. Rabaey, M. Pedram, and P. Landman, "Low Power Design Methodologies," *Kluwer*, Boston, 1995.

[36] P. P. Gelsinger, "Microprocessors for the new millennium: Challenges, opportunities, and new frontiers," *Proc. Intl. Symp. Circ. and Sys. ISCAS'01*, Sydney, Australia, May 2001, pp. 22–25.

[37] E. J. Nowak, "Maintaining the benefits of CMOS scaling when scaling bogs down," *IBM J. Res. & Dev.*, vol. 46, pp. 169–180, Mar./May 2002.

[38] S. Aunet, Y. Berg, O. Tjore, Ø. Næss, and T. Sæther, "Four-MOSFET floating-gate UV-programmable elements for multifunction binary logic," *Proc. World Multiconf. Sys. Cyber. & Informatics*, Orlando, FL, USA, Jul. 2001, vol. 3, pp. 141–144.

[39] T. Ytterdal, and S. Aunet, "Compact low-voltage self-calibrating digital floating-gate CMOS logic circuits," *Proc. Intl. Symp. Circ. and Sys. ISCAS'02*, Scottsdale, AZ, USA, May 2002, vol. 5, pp. 393–396.

[40] S. Aunet, T. Ytterdal, Y. Berg, and T. Sæther, "Multiple-input floating-gate linear threshold element tuned by well potential adjustment ," *Proc. Norchip Conf.*, Copenhagen, Denmark, Nov. 2002, pp. 220–225.

[41] Leiv Eiriksson Nyskaping, Trondheim, Snorre Aunet, Norwegian patent application no. 20035537, Dec. 2003

[42] S. Aunet, B. Oelmann, S. Abdalla, and Y. Berg "Reconfigurable subthreshold CMOS perceptron," *Proc. Intl. Joint Conf. Neural Networks IJCNN'04*, Budapest, Hungary, Jul. 2004, pp. 1983–1988.

[43] T. S. Lande, D. T. Wisland, T. Sæther, and Y. Berg "FLOGIC – Floating-gate logic for low-power operation," *Proc. Intl. Conf. Circ. Electr. Sys. ICCES'96*, Rhodos, Greece, Oct. 1996, vol. 2, pp. 1041–1044.

[44] T. Kobayashi, and T. Sakurai, "Self-adjusting threshold-voltage scheme (SATS) for low-voltage high-speed operation," *Proc. Custom Integr. Circ. Conf. CICC'94*, San Diego, CA, USA, May 1994, pp. 271–274.

[45] J. B. Lerch, "Threshold gate circuits employing field-effect transistors," U.S. Patent 3 715 603, Feb. 6, 1973.

[46] A. Weinberger, and J. L. Smith, "A logic for high-speed addition," *Natl. Bur. Stand.*, Circ. 591, pp. 3–12, 1958.

[47] P. M. Kogge, and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comp.*, vol. 22, Aug. 1973, pp. 786–793.

[48] R. E. Ladner, and M. J. Fischer, "Parallel prefix computations," *J. ACM*, vol. 27, Oct. 1980, pp. 831–838.

[49] R. P. Brent, and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comp.*, vol. 31, Mar. 1982, pp. 260–264.

[50] T. Han, D. A. Carlson, and S. P. Levitan, "VLSI design of high-speed, low-area addition circuitry," *Proc. Intl. Conf. Comp. Design ICCD'87*, 1987, 418–422.

[51] V. Beiu, "A survey of perceptron circuit complexity results," *Proc. Intl. Joint Conf. Neural Networks IJCNN'03*, Portland, OR, USA, Jul. 2003, vol. 2, pp. 989–994.

[52] P. Celinski, S. Al-Sarawi, D. Abbott, S. D. Cotofana, and S. Vassiliadis, "Logical effort based design exploration of 64-bit adders using a mixed dynamic-CMOS/threshold-logic approach," *Proc. Annual Symp. VLSI ISVLSI'04*, Lafayette, LA, Feb. 2004, pp. 127–132.

[53] A. Djupdal, S. Aunet, and V. Beiu "Ultra low power neural inspired addition: When serial might outperform parallel architectures," *Intl. Work-conf. Artif. Neural Networks IWANN'05*, Barcelona, Spain, Jun. 2005, under review.

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

[55] A. M. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Trans. VLSI Sys.*, vol. 10, Feb. 2002, pp. 20–29.

[56] S. Aunet, and V. Beiu, "Ultra low power fault tolerant neural inspired CMOS logic," *Intl. Joint Conf. Neural Networks IJCNN'05*, Montréal, Canada, Jul.-Aug. 2005, under review.