

On Using Recursive TMR as a Soft Error Mitigation Technique

Darshan D. Thaker[†]

Francois Impens[‡]

Isaac L. Chuang[‡]

Rajeevan Amirtharajah[†]

Frederic T. Chong[†]

[†] {ddthaker, ramirtha, ftchong} @ucdavis.edu

University of California at Davis

[‡] {impens, ichuang} @mit.edu

Massachusetts Institute of Technology

1 Introduction

As the minimum feature size of process technologies continues to decrease, microprocessor designers are faced with new reliability challenges. Feature sizes of less than $0.25\mu m$ result in an increased likelihood of noise-related faults that are the result of electrical disturbances in the logic values held in circuits and on wires [9, 6]. Natural radiation such as neutrons produced by cosmic rays and alpha particles generate electron-hole pairs as they pass through a semiconductor device. This may lead to transient faults that cause single bit upsets, which in turn may introduce a logical fault in the circuit. In addition, as the number of transistors increases, so does the complexity, which makes verification a much harder process, thus increasing the chance of undetected errors.

A considerable amount of recent research has focused on single errors due to the above factors, however it is unlikely that such events will always occur in isolation. It is the goal of the architect therefore to develop techniques to address transient faults due to simultaneous instances of single bit upsets.

It is known that reliable circuits can be constructed from error-prone components, but at the cost of increased circuit size and latency [10]. Until recently, this additional overhead did not justify the use of such fault-tolerant circuits. Now however, with smaller feature sizes pushing us towards unreliable components, should we be satisfied with large devices for the sake of reliability? Or can we devise techniques to increase the reliability of the smaller devices?

In this abstract, we start by showing that with a polynomial increase in resources, a reliable circuit can be built from unreliable components. We then talk about recursive triple modular redundancy (RTMR) techniques for building fault-tolerant circuits and consider noise models for CMOS devices where RTMR is beneficial and models for which it is deficient. Thereafter, we discuss micro-architecture approaches towards creating designs that tradeoff reliability, speed and area and show how RTMR could be used in these designs. We end with our plans for future work.

2 Fault-Tolerance

Modular redundant design was proposed by von Neumann [10] in a seminal paper, in 1956, at a time when the absence of reliable electronic components motivated the need for fault-tolerant designs. These techniques were not used in processor design since their overhead was beyond the component technologies of that time. Thereafter, due to the remarkable progress in the reliability of logic gates, there was little need to embed fault-tolerance in processor architecture. Winograd and Cowan [15] improved on von Neumann's work and obtained the following result:

Theorem 1: *A circuit containing N perfect gates can be simulated with probability of failure ϵ using $O(N \cdot \text{poly}(\log(N/\epsilon)))$ error-prone gates which fail with probability p provided $p < p_{th}$, where p_{th} is a constant threshold independent of N*

In order to protect the data against errors, the circuit consists of blocks that operate directly on data encoded in error correction codes. In this paper, we consider error correction codes in which each codeword represents a single logical bit, and where a gate can be transversally applied on the bits of the codeword. We use these codes, such as the triple redundancy code, to reduce error-propagation such that an error in any single gate does not lead to more than one error in each encoded output block, thus leading to what we call fault-tolerant procedures.

To prove the above theorem, we first prove the existence of p_{th} and start with $N = 1$. Let $p < \epsilon$. A circuit, C_N , consists of three blocks that fail with probability at most ϵ and one gate with failure probability p . C_N fails with a probability of at most $3\epsilon + p < 4\epsilon$. In the simplest, triple redundancy scheme, C_N can be made fault-tolerant by replacing it with three copies of C_N , the outputs of each of these becomes the input to a three-input majority voting gate. The new circuit has a failure probability of at most $3(4\epsilon)^2 + p$. Assuming $p < \epsilon^2$, the probability is bounded above by $49\epsilon^2$. In order for this circuit to be fault-tolerant, we need $49\epsilon^2 < \epsilon$, which gives $\epsilon < 1/49$ and

$$p < 1/(49)^2 \approx 0.0004$$

In order to obtain a circuit of size $O(N \text{poly}(\log(N/\epsilon)))$ that is fault-tolerant, we recursively perform the above circuit construction such that, we compute on encoded data, which itself is encoded and so on. Let c be a constant that describes the number of ways in which an encoded output can have more than one error in the circuit block. The failure probability at level one encoding is cp^2 . For two levels of recursion, this probability becomes $c(cp^2)^2$. Thus, the probability of failure at k levels of recursion is $\frac{(cp)^{2^k}}{c}$. An N gate circuit fails with probability at most ϵ if each gate fails has a failure probability of at most ϵ/N . Then k needs to satisfy

$$\frac{(cp)^{2^k}}{c} < \frac{\epsilon}{N}$$

As a result,

$$k = \left\lceil \log \left[\frac{\log(N/\epsilon c)}{\log(1/cp)} \right] \right\rceil$$

If d is the size of a fault-tolerant procedure in number of gates, the total size for a fault-tolerant circuit, N_{ft} , simulating N gates is $N_{ft} = Nd^k$. Using the value of k from above,

$$\begin{aligned} N_{ft} &= Nd \left\lceil \log \left[\frac{\log(N/\epsilon c)}{\log(1/cp)} \right] \right\rceil \\ &= N \cdot \text{poly}(\log(N/\epsilon)) \end{aligned} \quad (1)$$

Since c and d are constants, we now have the desired result.

3 Redundancy techniques

One of the guidelines of fault-tolerant procedures is that we always operate on encoded data. Consider the simple triple redundancy technique where we have three gates whose output is fed to a 3-majority voting gate. Clearly, this scheme violates the above guideline since the 3-majority gate decodes the data. Assume that we use NAND gates in our circuit. Such a design that does not decode the data can be built using d identical NAND gates and d identical d -majority gates. Figure 1 shows such a design for $d=3$. In this circuit, for a failure to occur, there can be either $\lceil d/2 \rceil$ NAND gate failures or $\lceil d/2 \rceil$ majority gate failures. For both of these there are $\binom{d}{t}$ possible choices. If the inputs to the NAND gates are correct, the probability of failure for this encoded NAND is at most:

$$p_{fail} \leq \binom{d}{t} p_1^t + \binom{d}{t} p_2^t \quad \text{with} \quad t = \left\lceil \frac{d}{2} \right\rceil$$

Here p_1 and p_2 are the failure probabilities of the NAND gate and the d -majority gate respectively. Let, $p_1 = p_2 = p$, which gives us $p_{fail} \leq 6p^2$. The threshold probability of an encoded NAND using the current design is $p_{th} = \frac{1}{6}$.

However, such a NAND gate will not exist independently and thus we have to consider its reliability in the presence of noisy inputs. Since we consider an encoded NAND taking two inputs issued by previous similar gates. The encoded computation is immune to $\lfloor \frac{d}{2} \rfloor$ or less failures or input deviations. From details described in [4], let the error probabilities of the NAND

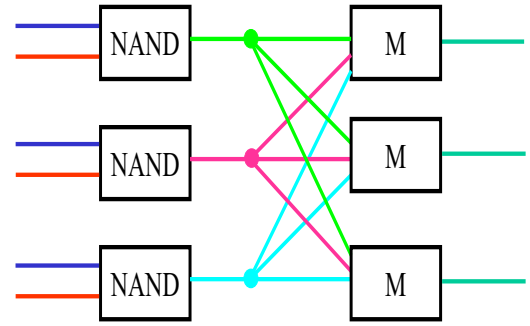


Figure 1: Distributed Majority Voting

Figure 2: Distributed Majority Voting

gate and the majority gate be bounded by p , the probability of failure of the encoded NAND is at most (for $d = 3$ and $t = 2$):

$$p_{fail} \leq c_3 p^2 \quad \text{with} \quad c_3 = 24 \quad (3)$$

This new bound corresponds to a threshold $p_{th} = \frac{1}{c_3} = \frac{1}{24}$. This result shows that NAND gates in a noisy circuit and encoded in the triple redundancy code, fail with a probability $O(p^2)$.

Since we are trying to build a reliable architecture out of unreliable circuits we are faced with a problem. While recursive voting leads to a double exponential decrease in the failure probability of the circuit, it also is hampered because a single error in the last d -majority gate can lead to an incorrect result. The solution is to combine recursive majority voting and multiplexing.

A multiplexed NAND gate at recursion level 2 can be obtained by modifying figure 1 in the following manner: Replace each NAND by its multiplexed version. In order for this to be fault-tolerant, we do it three times in parallel, triplicating each output of the multiplexed NAND gates. The final output is the reunion of all 3-majority gates. Such recursive, multiplexed circuits of level k perform encoded computations on the triple redundancy code concatenated k times with itself.

3.1 Resource usage for recursive multiplexing

Let $N_1(k)$ and $N_2(k)$ denote respectively the numbers of NAND and 3-majority gates in a multiplexed, encoded NAND of level k . Due to recursive multiplexing, $N_1(k) = 3N_1(k-1)$ and $N_2(k) = 3N_2(k-1) + 3^k k$. We can start with $N_1(0) = 1$ and $N_2(0) = 0$ to arrive at $N_1(k) = 3^k$ and $N_2(k) = 3^k k$. If we assume that 3-majority gates and NAND gates are built with the same technology, the circuit for a level k multiplexed NAND gate requires $N(k) = 3^k(k+1)$ devices. Reliability of such a NAND gate with perfectly encoded inputs is $p_1^{(k)} \leq \frac{1}{c} (cp)^{2^k}$ with $c = 6$ and with imperfect inputs is given by $p_1^{(k)} \leq \frac{1}{c_3} (c_3 p)^{2^k}$ with $c_3 = 24$. The interested reader can look at [4] for details.

4 Results

Our goal is to construct a circuit that fails with probability at most ϵ . The simplest approach is to enlarge small devices until they consume an area A such that $p_{fail}(A) \leq \epsilon$. To compare this against fault-tolerant constructions, we consider devices whose failure probabilities can be approximated by $p_{fail}(A) = 1/A^\gamma$, where γ is a parameter of the technology. (Lack of space restricts us from showing results for other device categories) We wish to determine if using RTMR benefits us in terms of area consumed. Let r represent the resource gain, i.e decrease in area usage, due to an RTMR design.

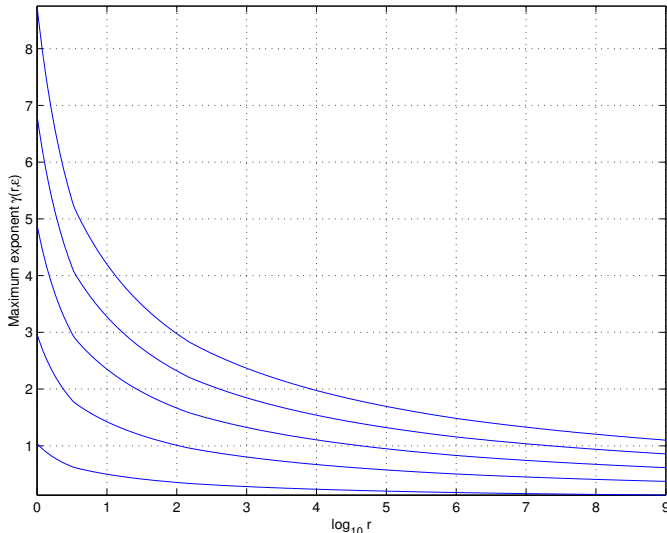


Figure 3: Maximum exponent $\gamma(r, \epsilon)$ for devices with $p(A) = A^{-\gamma}$ as a function of the resource gain r for different reliability requirements $\epsilon = 10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}, 10^{-15}$. Upper curves correspond to higher reliability requirements.

4.1 Class A

Energetic particles, like neutrons generated by from cosmic rays and alpha particles emitted by packaging materials give rise to single-event upsets when passing through semiconductor devices. FIT (Failures in Time) is a term used by microprocessor designers when estimating soft error rates and is provides a measure of the degree of reliability a circuit needs to maintain error-free operation. Cohen et al. [9] have studied the change in the effect of α particles on FIT rates of dynamic circuits with scaling in CMOS technology. They conducted experiments where circuits were directly exposed to a α -particle emitting Th^{232} foil. Their results indicate that $p_{fail}(A) \approx A^{-2.5}$.

Now recall that γ is a parameter of the technology and r is resource gain. Also, let the exponent $\gamma(r, \epsilon)$ be the maximum exponent for which r can be achieved at reliability ϵ . Figure 4 shows the resource gains that can be achieved for class A devices. We can see that for a device that fails as $p_{fail}(A) = 1/A$, a resource gain of 100 can be achieved for

$\epsilon = 10^{-6}$. When the failures in a circuit are dominated by errors due to energetic particle strikes, using smaller, less reliable components in a RTMR design results in significant resource savings compared to using larger, more reliable devices.

5 Microarchitecture directions

It is clear from the preceding sections that in order to build processors that are immune to soft errors, designers cannot solely rely on error-correcting circuits. In fact, The last few years have seen a number of approaches to building fault-tolerant microprocessors. In many approaches [12, 14] redundant hardware or redundant threads are used to process instructions and their results compared to ensure reliability. A more recent paper [5] staggers redundant threads in a superscalar pipeline to reduce issue queue bandwidth constraints.

Studies have also been done to determine what components of the micro-architecture are the most vulnerable to soft errors. In [13], the authors define AVF (architecture vulnerability factor) of a structure as the probability that a fault in that particular structure will result in an error. In another study, [11] a detailed latch-accurate RTL simulation of an Alpha processor was conducted. Faults were injected into the simulation to study how they propagate and become errors at the micro-architecture level. In this way, the most vulnerable components of the micro-architecture were determined and then protected using light-weight error-correction mechanisms.

5.1 DIVA

One clever approach is the DIVA micro-architecture [1] where a conventional superscalar pipeline is augmented by a simpler checker pipeline. Before the completed instructions in the main out-of-order pipeline are committed, they proceed in program order to the checker pipeline, accompanied by their computed results. Each instruction is recomputed in the checker and if a mismatch is found, an exception is raised which results in the main pipeline resuming execution at the offending instruction.

To offset this, the DIVA checker is a very simple design, making it easier to verify and thus more reliable. When we consider small feature sizes where errors are primarily caused by noise and energetic particles, there are scenarios where the DIVA architecture does not provide adequate reliability. Recollect our RTMR designs in the previous sections that always encode all computation and assume that the underlying devices were unreliable. If we use devices that are small enough to yield benefits in terms of performance and power, and yet good enough that we can operate just beyond the reliability threshold, we can use the DIVA idea to safeguard against all manners of transient errors in the following manner: let the main pipeline be constructed as normal and the DIVA checker should be built with our RTMR designs. Since the original checker only consumed 6% of the area of the core processor, using level 1 recursion for the checker still keeps its area well below that of the core. Figure 5.1 shows the increase in overall size of the checker with the increase in the number of checker

components protected using the RTMR design.

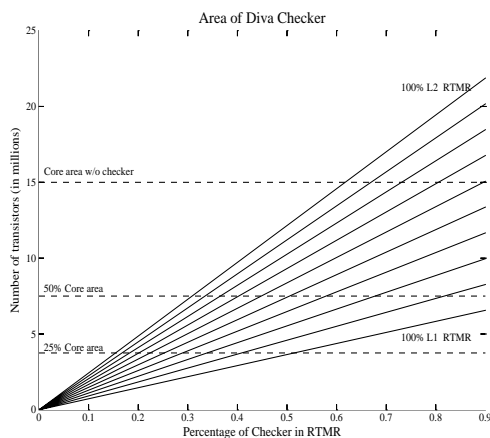


Figure 4: Increase in area of the DIVA checker with increasing use of RTMR. The bottom most solid line is 100% Level 1 RTMR, the one above it is (90% L1+ 10% L2) and so on

6 Future Directions

In the previous section we described a micro-architecture design that combined the DIVA architecture and RTMR to detect and correct errors. For this design, we need to address the issues of speed of the checker processor with respect to the main pipeline, and power requirements of the modified checker.

As processor feature sizes move into the nanoscale region, voltage scaling will become the dominant factor in leading to single bit-flips [9]. This means that we need to conduct in detailed studies of which components of a microprocessor are the most susceptible to errors due to aggressive voltage scaling, and offer remedies to alleviate the vulnerability of these components. We plan on studying this issue using a tool like the Liberty Simulation Environment [8]

With the increase in the variety of nanoscale components like carbon nanotubes, silicon nanowires and SETs, researchers have begun to explore how these devices could be used to build computational circuits [2, 7]. Others have dwelt on the fact that these nanoscale devices are inherently unreliable and devised defect-tolerant techniques [3] to build reliable circuits. What is missing however, is a study of the trade-offs between area, reliability and performance when different nanoscale devices are used to build circuits.

References

[1] C.Weaver and T.Austin. A fault tolerant approach to microprocessor design. In *Dependable Systems and Networks*, 2001.
 [2] A. DeHon, P. Lincoln, and J.E. Savage. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Transactions on Nanotechnology*, 2:165–174, 2003.

[3] J. Han and P. Jonker. A system architecture solution for unreliable nanoelectronic devices. *IEEE Trans. on Nanotechnology*, 1(4):201–208, 2002.
 [4] F. Impens. Fine-grained fault-tolerance: Reliability as a fungible resource. *PhD thesis, Massachusetts Institute of Technology*, 2004.
 [5] J.C.Smolens, J.Kim, J.C.Hoe, and B.Falsafi. Efficient resource sharing in concurrent error detecting superscalar microarchitectures. In *International Symposium on Microarchitecture (MICRO)*, 2004.
 [6] J.F.Ziegler. Terrestrial cosmic rays. *IBM Journal of Research and Development*, 40(1):19–39, January 1996.
 [7] L.J.K.Durbeck and N.J.Marcias. The cell matrix: An architecture for nanocomputing. *Nanotechnology*, 12:217–230, 2001.
 [8] M.Vachharajani, N.Vachharajani, D.A.Penry, J.A.Blome, and D.I.August. Microarchitectural exploration with liberty. In *International Symposium on Microarchitecture (MICRO)*, 2002.
 [9] N.Cohen, T.S.Sriram, N.Leland, D.Moyer, S.Butler, and R.Flatley. Soft error considerations for deep-submicron cmos circuit applications. *IEEE International Electron Devices Meeting: Technical Digest*, pages 315–319, December 1999.
 [10] J. Von Neumann. Probabilistic logic and the synthesis of reliable organisms from unreliable components. *Automata Studies, Ann. of Math. Studies*, 34:43–98, 1956.
 [11] N.J.Wang, J.Quek, T.M.Rafacz, and S.J.Patel. Characterizing the effects of transient faults on a high-performance processor pipeline. In *International Symposium on Microarchitecture (MICRO)*, 2003.
 [12] S.K.Reinhardt and S.S.Mukherjee. Transient fault detection via simultaneous multithreading. In *International Symposium on Computer Architecture*, 2000.
 [13] S.S.Mukherjee, C.Weaver, J. Emer, S.K.Reinhardt, and T.Austin. A systematic methodology to compute the architectural vulnerability factors for a high performance microprocessor. In *International Symposium on Microarchitecture (MICRO)*, 2003.
 [14] T.N.Vijaykumar, I.Pomeranz, and Karl Cheng. Transient fault recovery using simultaneous multithreading. In *International Symposium on Computer Architecture*, 2002.
 [15] S. Winograd and J.D. Cowan. *Reliable Computation in the Presence of Noise*. MIT Press, Cambridge, Massachusetts, 1963.