

Recursive TMR: Scaling Fault Tolerance in the Nanoscale Era

Darshan D. Thaker

University of California, Davis

Rajeevan Amirtharajah

University of California, Davis

Francois Impens and Isaac L. Chuang

Massachusetts Institute of Technology

Frederic T. Chong

University of California, Santa Barbara

Editor's note:

Although recursive voting leads to a double exponential decrease in a circuit's failure probability, a single error in the last majority gate can cause an incorrect result, hampering the technique's effectiveness. Combining recursive majority voting and multiplexing helps alleviate this problem. However, the cost of this fault-tolerant approach might not be worthwhile, depending on the nature of the faults.

—R. Iris Bahar, Brown University

from error-prone components—but at the cost of increased circuit size and latency.⁴ Until recently, this additional overhead did not justify the use of such fault-tolerant circuits. Now, however, with denser feature sizes pushing us toward unreliable components, should we be satisfied with large devices for the sake of reliability? Or can we devise techniques to increase the smaller devices'

AS PROCESS TECHNOLOGIES DECREASE in feature size, designers face new reliability challenges. Feature sizes of less than 0.25 μm increase the risk of noise-related faults that result from electrical disturbances in the logic values held in circuits and on wires.^{1,2} Such transient faults can cause single-bit upsets, which in turn can introduce a logical fault in the circuit. Natural radiation, such as neutrons and alpha particles, is an increasingly prominent cause of transient faults.² This radiation's inherent randomness makes protecting against transient errors a complex task. Denser feature sizes encourage designs with hundreds of millions of transistors. Designers use many of these transistors for complex components such as dynamic schedulers and use others in simpler structures such as caches.

As overall complexity increases, it places a greater burden on verification techniques, leading to an increased chance of undetected errors. In addition, the raw error rate per latch or SRAM bit will remain constant even as their sizes decrease.³ To make good use of the cost and speed advantages of new technologies, designers must address these reliability concerns.

We know that reliable circuits can be constructed

reliability? Intuitively, it seems that a scalable approach would employ fault-tolerant logic rather than assume that all devices will remain error free throughout a computation. Yet fault-tolerant logic comes at a cost that we must carefully consider and compare against simply using larger, more reliable devices.

Conventional wisdom suggests that smaller is better and that fault-tolerant circuit techniques can compensate for decreasing reliability. We find that this is not necessarily true for CMOS devices using the most asymptotically efficient fault-tolerant circuits known—recursive triple modular redundancy (RTMR) circuits. In this article, we classify the sources of noise that can be scalably corrected (where using RTMR is beneficial as device size scales) with RTMR and those that cannot. In particular, we have found that single-event upsets caused by energetic particles can be effectively compensated with RTMR. Flicker noise in devices, however, is not competitively correctable. In other words, noise models show that an RTMR circuit composed of small, less reliable devices does not always compete in speed and area with an equivalent circuit composed of

larger, more reliable devices. In light of this finding, we discuss microarchitectural design options for mixing large and small devices to trade off reliability, speed, and area.

Fault tolerance

In a seminal 1956 paper, when the absence of reliable electronic components motivated the need for fault-tolerant designs, von Neumann proposed modular redundant design.⁴ Yet von Neumann's techniques were not used in processor design because their overhead was beyond the capacities of component technologies of the time. Later, because of the remarkable progress in logic gate reliability, there was little need to embed fault tolerance in processor architectures. In 1963, Winograd and Cowan improved on von Neumann's work and presented the following theorem:

A circuit containing N error-free 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 .⁵

Figure 1 shows the essential idea behind the theorem: To obtain an error-free gate, use multiple copies of G and feed their outputs to majority gate M . The failure probability of each gate on the left is p , and the circuit on the left fails with probability ϵ .

We assume that the circuit used consists of blocks that operate directly on data encoded in error-correcting codes. In this article, we consider error-correcting codes in which each word represents a single logical bit and a gate can be transversally applied on bits of the word. We use these codes, such as triple-redundancy code, to reduce error propagation such that an error in any single gate causes no more than one error in each encoded output block. Thus we obtain what we call fault-tolerant procedures.

To prove Winograd and Cowan's theorem, we first prove the existence of p_{th} . We start with $N=1$. Consider circuit C_N that consists of three blocks with a failure probability of at most ϵ and one gate with a failure probability of p . Let $p < \epsilon$. C_N fails with a probability of at most $3\epsilon + p < 4\epsilon$. The simplest triple-redundancy scheme can make C_N fault tolerant by replacing it with three copies of C_N , the outputs of each becoming the input to a three-input majority-voting gate. This new circuit has a failure probability of at most $3(4\epsilon)^2 + p$. Assuming $p < \epsilon^2$, the probability is bounded above by $3(16\epsilon^2) + \epsilon^2 = 49\epsilon^2$. For

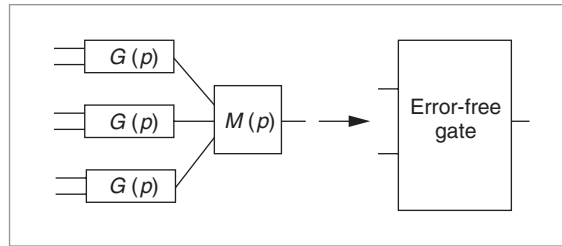


Figure 1. Using imperfect gates (left) to simulate an error-free gate (right). Each imperfect gate has a failure probability p , where $p < p_{th}$, and the total failure probability of the circuit on the left is ϵ .

this circuit to be fault tolerant, we need $49\epsilon^2 < \epsilon$, which gives $\epsilon < 1/49$ and $p < 1/(49)^2 \approx 0.0004$.

To obtain a fault-tolerant circuit of size $O(N \text{poly}(\log(N/\epsilon)))$, we recursively perform the circuit construction just described. Using two recursion levels, we construct an encoded circuit that computes on encoded data. 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. The failure probability at level-one encoding is cp^2 . For two levels of recursion, this probability becomes $c(cp^2)^2$. In this manner, the failure probability of a circuit encoded at k levels of recursion is

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

A circuit of N gates fails with a probability of at most ϵ , if each gate that fails has a failure probability of at most ϵ/N . To achieve this probability, k must 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 of a fault-tolerant circuit N_t simulating N gates is $N_t = Nd^k$. Using the value of k just given,

$$N_t = Nd^{\left\lceil \log \left[\frac{\log(N/\epsilon c)}{\log(1/cp)} \right] \right\rceil} = N \text{poly}(\log(N/\epsilon))$$

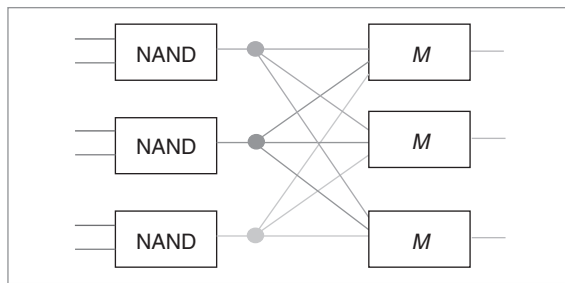


Figure 2. Distributed majority voting.

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

Redundancy techniques

A guideline for using fault-tolerant procedures is that computation should always be performed on encoded data to eliminate the possibility of errors corrupting data in an unencoded state. Consider the simple triple-redundancy technique in which we have three gates whose outputs are fed to a three-input majority-voting gate.⁶ Clearly, this scheme violates the guideline, because the three-majority gate decodes the data. We can build a design that does not decode data using d identical NAND gates and d identical d -majority gates.

Figure 2 shows such a circuit for $d = 3$. Here, either $\lceil d/2 \rceil$ NAND gate failures or $\lceil d/2 \rceil$ majority-gate failures are required for the complete circuit to fail. In both cases, there are $\binom{d}{t}$ possible choices. If the inputs to the NAND gates are correct, the probability of failure for this encoded NAND gate is at most

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

where p_1 and p_2 are the failure probabilities of the NAND gate and the d -majority gate, respectively. Assuming $p_1 = p_2 = p$, we get $p_{\text{fail}} \leq 6p^2$. As a result, the threshold probability of an encoded NAND gate using the current design is $p_{\text{th}} = 1/6$. For the rest of this article, we assume that our circuits contain only NAND gates.

A NAND gate like the one just described will never exist independently. Therefore, we consider its reliability in the presence of noisy inputs. From Figure 2 we can see that

- If $\lceil d/2 \rceil = 2$ or more wires feeding the d -majority gate deviate, an error in the output is highly probable.
- If $\lceil d/2 \rceil = 1$ or fewer wires feeding the d -majority gate

deviate and the output is corrupt by one bit, the error is from the d -majority gate.

We now consider an encoded NAND gate taking two inputs issued by similar gates. The encoded computation is immune to $\lfloor d/2 \rfloor$ or fewer failures or input deviations. When $t = \lceil d/2 \rceil$ failures or input deviations lead to an incorrect output, the causes could be one of the following:

- There were t failures in the d -majority gate. The number of such possibilities is $\binom{d}{t}$.
- There were k failures in the NAND gates and $t - k$ deviations in the inputs for $k = 1, \dots, \lceil d/2 \rceil$. The number of possible events in this case are $\binom{d}{k} 2^{t-k} \binom{d-k}{t-k}$.
- There were t deviations in the input. Each deviating input fed a different NAND gate, and these t deviations affected both input code words simultaneously. There are $2^{\binom{d}{t}} - 2^{\binom{d}{t}}$ such possible events.

The error probabilities of the NAND gate and the majority gate are bounded by p . Therefore, the probability of failure for the encoded NAND gate is at most (for $d = 3$ and $t = 2$)

$$p_{\text{fail}} \leq \binom{3}{2} p^2 + 2 \binom{3}{1} \binom{2}{1} p^2 + \binom{3}{2} p^2 + (4 - 2) \binom{3}{2} p^2 \leq c_3 p^2 \text{ with } c_3 = 24$$

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

Recursive multiplexing

Because we are trying to build a reliable architecture out of unreliable circuits, we face a problem: Although recursive voting leads to a double exponential decrease in the circuit's failure probability, the technique is faulty because a single error in the last d -majority gate can cause an incorrect result. The solution is to combine recursive majority voting and multiplexing.

Figure 3 shows a multiplexed NAND gate at recursion level 2. We obtain this by modifying Figure 2 as follows: We replace each NAND gate with its multiplexed version. Let the outputs of the upper multiplexed NAND gate be $b_{1,1}$, $b_{1,2}$, and $b_{1,3}$, and the outputs of the second multiplexed NAND gate be $b_{2,1}$, $b_{2,2}$, and $b_{2,3}$. Finally, $b_{3,1}$, $b_{3,2}$, and $b_{3,3}$ are the outputs of the last multiplexed NAND gate. Now we perform 3-majority voting on $b_{1,1}$,

$b_{1,2}$, and $b_{1,3}$ on $b_{2,1}$, $b_{2,2}$, and $b_{2,3}$ and on $b_{3,1}$, $b_{3,2}$, and $b_{3,3}$. To make this circuit fault tolerant, we perform the multiplexing three times in parallel, triplicating each output of the multiplexed NAND gates. The final output is the reunion of all the 3-majority gates. Such recursive, multiplexed circuits of level k perform encoded computations on triple redundancy code concatenated k times with itself.

Resource use for recursive multiplexing

Let $N_1(k)$ and $N_2(k)$ denote respectively the numbers of NAND gates and 3-majority gates in a multiplexed, encoded NAND of level k . Because of 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 3-majority gates and NAND gates are built using the same technology, the circuit for a level- k multiplexed NAND gate requires $N(k) = 3^k(k + 1)$ devices. The reliability of such a NAND gate with perfectly encoded inputs is

$$p_1^{(k)} \leq \frac{1}{c} (cp)^{2^k}$$

where $c = 6$. With imperfect inputs, its reliability is

$$p_1^{(k)} \leq \frac{1}{c_3} (c_3 p)^{2^k}$$

where $c_3 = 24$.

Results

To understand how to use these fault-tolerant constructions effectively, we study different models for device failure probability. Our goal is to construct a circuit that fails with a probability of at most ϵ . The simplest approach is to enlarge small devices until they consume an area A such that $p_{\text{fail}}(A) \leq \epsilon$. To compare this construction against fault-tolerant constructions, we divide the smaller devices' failure probabilities into separate classes. Devices whose failure probabilities can be approximated by $p_{\text{fail}}(A) = 1/A^\gamma$ occupy class A, and devices that exhibit a failure probability of $p_{\text{fail}}(A) = (1/2)\text{erfc}(A^\gamma)$ occupy class B (γ is a parameter of the technology). For both classes, we wish to determine if using RTMR yields benefits in terms of area consumed.

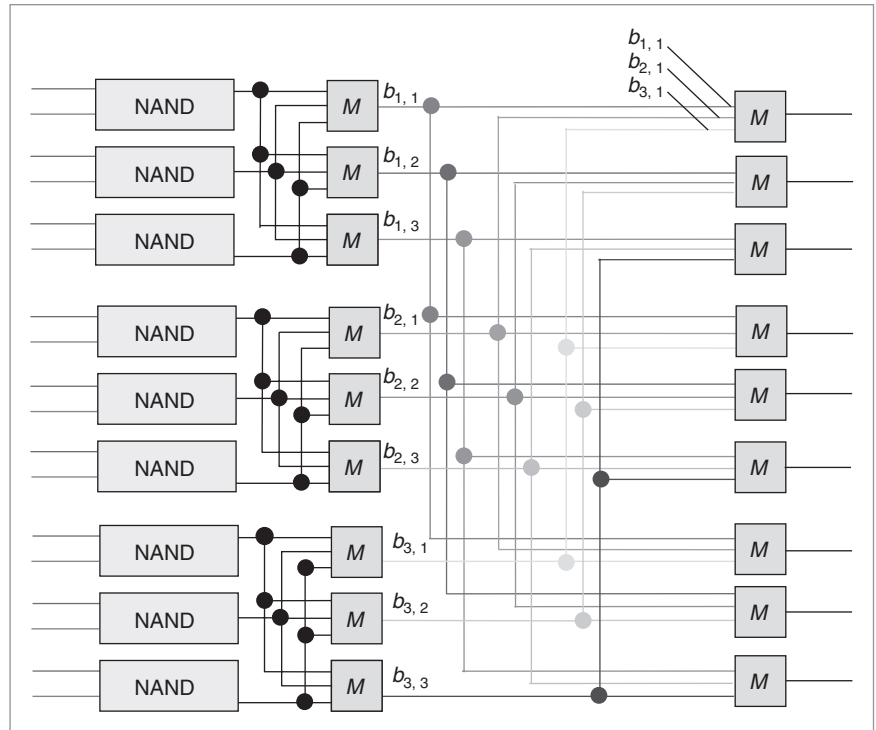


Figure 3. Multiplexed NAND gate at recursion level 2.

Class A

Energetic particles, such as neutrons generated by cosmic rays and alpha particles emitted by packaging materials, give rise to single-event upsets when passing through semiconductor devices. Logic errors caused by bit flips from single-event upsets are called *soft errors*. *Failures in time* (FIT) is a term microprocessor designers use in estimating error rates. FIT provides a measure of the degree of reliability a system requires to maintain error-free operation. Cohen et al. have studied the changing effects of alpha particles on FIT rates of dynamic circuits with scaling in CMOS technology.¹ They conducted experiments in which they directly exposed circuits to an alpha-particle-emitting Th²³² foil. To measure failures caused by soft errors, they set all cells to specified logic levels, exposed the chip for fixed time intervals under specific voltage conditions, and checked for corrupted logic states. They repeated this for different device sizes and then used simulations to obtain FIT rates for very small CMOS feature sizes. Their results show that $p_{\text{fail}}(A) \approx A^{-2.5}$.

Recall that γ is a technology parameter, and let r represent resource gain—that is, decrease in area use resulting from an RTMR design. Also, let the exponent $\gamma(r, \epsilon)$ be the maximum exponent for which r is achiev-

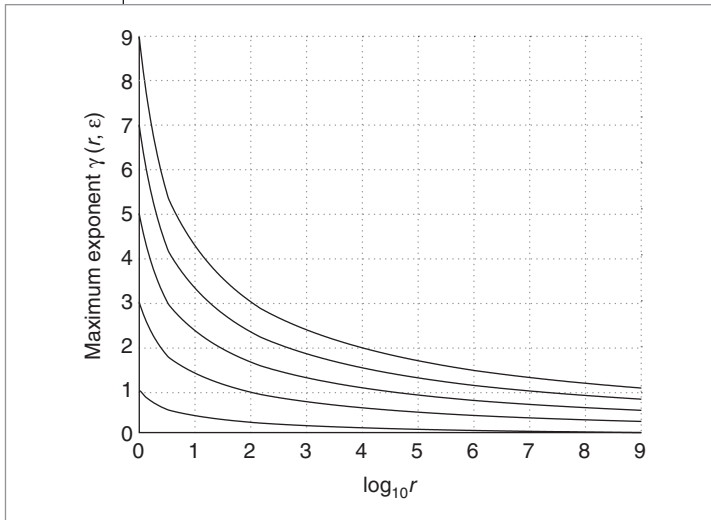


Figure 4. Maximum exponent $\gamma(r, \epsilon)$ for class A, $p(A) = A^{-\gamma}$, as a function of resource gain r for reliability requirements $\epsilon = 10^{-3}$, 10^{-6} , 10^{-9} , 10^{-12} , and 10^{-15} . Upper curves correspond to higher reliability requirements.

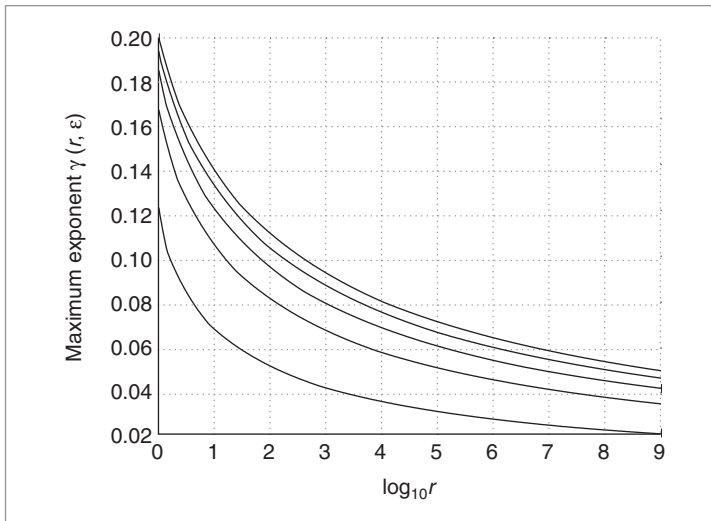


Figure 5. Maximum exponent $\gamma(r, \epsilon)$ for class B, $p(A) = (1/2)\text{erfc}(A^\gamma)$, as a function of resource gain r for reliability requirements $\epsilon = 10^{-3}$, 10^{-6} , 10^{-9} , 10^{-12} , and 10^{-15} . Upper curves correspond to higher reliability requirements.

able at reliability ϵ . Figure 4 shows the resource gains achievable by class A devices. We can see that a device with failure probability $p_{\text{fail}}(A) = 1/A$ achieves a resource gain of 100 for $\epsilon = 10^{-6}$. When errors resulting from energetic particle strikes dominate a circuit's failures, using smaller, less reliable components in an RTMR design provides more-significant resource savings than simply using larger, more reliable devices.

Class B

To study the effects of noise on a CMOS circuit's reliability, we use Sarpeshkar, Delbrück, and Mead's noise resource equation,⁷ which relates a transistor's total input-referred noise power to its area and the power it dissipates:

$$v_n^2 = \frac{K_w(p)}{I^p}(f_h - f_l) + \frac{K_f}{A} \ln(f_h / f_l)$$

The first term is shot noise, and the second term describes flicker ($1/f$) noise. Also, $K_f = B/C_{\text{ox}}$ is the ratio of B , a measure of impurities or defects in the gate oxide, and C_{ox} , the gate oxide capacitance. A is the device area and $f_h - f_l = \Delta f$ is the frequency bandwidth. Let the probability of noise voltage be distributed as a Gaussian distribution:⁷

$$p_V(v) = \frac{1}{\sqrt{2\pi v_n^2}} e^{-v^2/2v_n^2}$$

Then, the probability that a dynamic circuit will fail owing to a bit flip caused by noise is

$$\begin{aligned} p_{\text{fail}} &= 2 \int_{v_{\text{th}}}^{\infty} p_V(v) dv \\ &= \sqrt{\frac{2}{\pi v_n^2}} \int_{v_{\text{th}}}^{\infty} e^{-v^2/2v_n^2} dv \\ &= \frac{2}{\sqrt{\pi}} \int_{x_{\text{th}}}^{\infty} e^{-x^2} dx \\ &= \text{erfc}\left(\frac{v_{\text{th}}}{\sqrt{2}v_n}\right) \end{aligned}$$

In these equations, $x_{\text{th}} = v_{\text{th}}/(\sqrt{2}v_n)$, and we assume threshold voltage v_{th} is not a function of area. This gives us a failure probability of $p_{\text{fail}}(A) = (1/2)\text{erfc}(A^\gamma)$ if we consider flicker noise the prominent cause of failure. Figure 5 shows the achievable resource gains. For a resource gain of $r = 100$ for $\epsilon = 10^{-6}$, an exponent γ below 0.08 is necessary. This reveals that a fault-tolerant design will be effective only for very low maximum exponents.

The γ value shows how a device's reliability will change with a change in area. As device sizes shrink, a technology with a smaller γ is more likely to suffer from errors than one with a larger γ . Depending on the primary cause of errors, at a small γ , using an RTMR design with smaller devices can be more efficient than simply using a larger, more reliable device. Table 1 shows the γ values below which the RTMR construction consumes

less area for both device classes, when $\epsilon = 10^{-9}$.

We conclude that when soft errors (class A) dominate, a greater range of values and thus technologies will benefit from employing RTMR and using smaller, less reliable devices.

Microarchitecture directions

Clearly, to build processors immune to soft errors, designers cannot rely solely on error-correcting circuits. In fact, in the past few years, researchers have offered several approaches to building fault-tolerant microprocessors.

Redundant threads

Some schemes use redundant hardware or redundant threads to process instructions and compare the results to ensure reliability.⁸ The redundant-thread approach improves on redundant-hardware performance, with one thread prefetching cache misses and computing branch outcomes for other threads. A more recent approach staggers redundant threads in a superscalar pipeline to reduce issue queue bandwidth constraints.⁹ Its main goal is to provide fault tolerance with minimum changes to existing designs without sacrificing performance.

DIVA

Another clever approach is the dynamic implementation verification architecture (DIVA), which augments a conventional superscalar pipeline with a simpler checker pipeline. Before the core processor commits the completed instructions in the main out-of-order pipeline, the instructions proceed in program order to the checker pipeline, accompanied by their computed results. The checker recomputes each instruction and, if it finds a mismatch, raises an exception, resulting in the main pipeline's resuming execution at the offending instruction. Figure 6 shows the DIVA structure.

Modern microprocessor designs are increasingly complex, making validation an extremely difficult process. To offset this, the DIVA checker is a very simple design, simplifying verification and improving reliability. Consider devices with small feature sizes, in which noise and energetic particles are the primary causes of errors. In some scenarios, the DIVA does not provide adequate reliability because the checker is less resistant to failure. Our RTMR designs in the previous

Device class	RTMR efficiency range for $\epsilon = 10^{-9}$
A	γ below 5.0
B	γ below 0.185

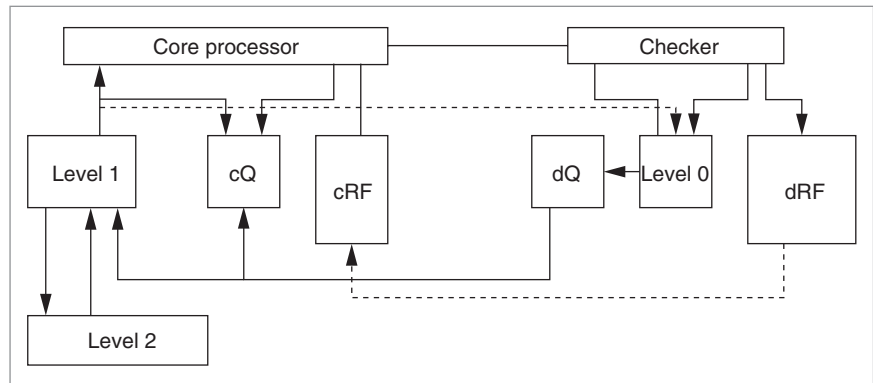


Figure 6. DIVA: cQ and dQ are store queues for the core and checker, respectively; cRF and dRF are the corresponding register files.

sections always encode computation and assume that the underlying devices are unreliable. However, if we use devices small enough to yield performance and power benefits, yet good enough to operate just beyond the reliability threshold, we can use the DIVA idea to safeguard against all types of transient errors. We can do this by constructing the main pipeline in the normal manner and building the DIVA checker with our RTMR design. Because the original checker consumed only 6% of the core processor's area, using level-1 recursion for the checker still keeps its area well below that of the core. Figure 7 shows the increase in the checker's overall size with the increase in the number of checker components protected using the RTMR design.

It might seem that the DIVA checker (without RTMR) should be slower than the core pipeline, but Weaver and Austin show that the checker with a four-wide pipeline is fast enough to keep up with the core.¹⁰ Adding RTMR to the checker will slow it down if we are restricted by logic speed, but if wire delay becomes the dominant factor at very small feature sizes, RTMR won't slow the checker.

Microarchitecture vulnerabilities

Recently, studies have attempted to determine which microarchitecture components are most vulnerable to soft errors. Mukherjee et al. defined a structure's architecture vulnerability factor (AVF) as the

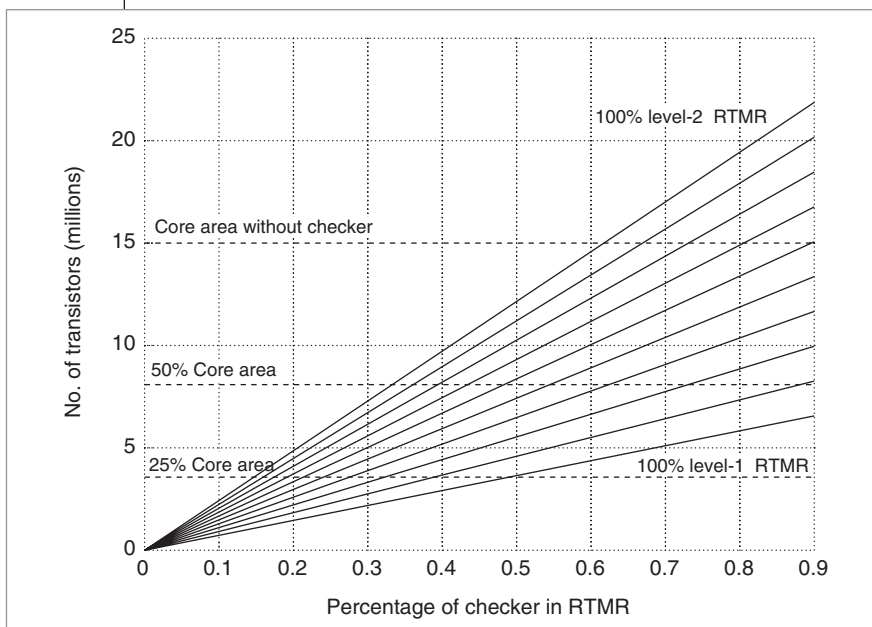


Figure 7. Increase in area of DIVA checker with increasing use of RTMR. The bottommost solid line is 100% level-1 RTMR; the one above it is 90% level-1 + 10% level-2 RTMR, and so on.

probability that a fault in the structure will result in an error.¹¹ They used detailed models of an IA64 processor to show that the AVF is 28% for instruction queues and 9% for execution units, across the CPU2000 benchmark suite.

In another study, Wang et al. conducted a detailed, latch-accurate RTL simulation of an Alpha processor.¹² They injected faults into the simulation to study how the faults propagate and become errors at the microarchitecture level. Thus, the researchers determined the microarchitecture's most vulnerable components and then protected them with lightweight error-correcting mechanisms. The goal of this work was to reduce errors as much as possible while keeping the overhead low. Because the goal of an error-correcting code is to safeguard against soft errors, the ECC logic itself needs protection. Employing RTMR would address this need.

Future directions

We have described microarchitecture designs that help overcome soft errors. In the future, however, as the number of transient errors increases, these approaches will be inadequate. Let's consider the DIVA approach. If the core pipeline is unmodified and we use RTMR to make the checker reliable, we can have scenarios in which the core frequently computes an incorrect result and we must restart the pipeline. Thus, we'll have a reli-

able processor at the cost of severe performance degradation. This also applies to redundant-thread approaches.

Architectures using lightweight constructions to protect vulnerable components might not experience performance hits but might suffer an increase in errors as single-event upsets overwhelm the protection mechanisms. This naturally leads us to a design using a DIVA approach, with the checker implemented completely in RTMR and the most vulnerable components of the core pipeline also protected by RTMR. In such an architecture, we would need to address how RTMR affects the system's overall speed.

With the increase in the variety of nanoscale components, such as carbon nanotubes, silicon nanowires, and single-electron transistors, researchers have begun to explore using these devices to build computational circuits.¹³ Others,

dwelling on the inherent unreliability of these nanoscale devices, have devised defect-tolerant techniques to build reliable circuits.¹⁴ Missing, however, is a study of the trade-offs between area, reliability, and performance when different nanoscale devices are used to build circuits.

DECREASING DEVICE SIZES have been the mainstay of faster, cheaper, and denser computation. As we enter the nanoscale era, however, system reliability becomes increasingly important. We have found that RTMR effectively compensates for the dominant sources of near-term noise, single events caused by energetic particles. On the other hand, noise sources associated with devices scale too quickly with device size, and RTMR circuits become too large and slow, so circuits made with larger, more reliable devices might be more competitive. Microarchitectural research is critical to the effective use of future nanoscale technologies. In particular, designers must develop methodologies for assessing future systems' error vulnerability and carefully apply large devices or fault-tolerant circuits to the areas of vulnerability. ■

References

1. N. Cohen et al., "Soft Error Considerations for Deep-

Submicron CMOS Circuit Applications," *IEEE Int'l Electron Devices Meeting Tech. Digest (IEDM 99)*, IEEE Press, 1999, pp. 315-319.

2. J.F. Ziegler, "Terrestrial Cosmic Rays," *IBM J. Research and Development*, vol. 40, no. 1, Jan. 1996, pp. 19-39.
3. P. Hazucha and C. Svenson, "Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate," *IEEE Trans. Nuclear Science*, vol. 47, no. 6, Dec. 2000, pp. 2586-2594.
4. J. von Neumann, "Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies*, C. Shannon and J. McCarthy, eds., Princeton Univ. Press, 1956, pp. 43-98.
5. S. Winograd and J.D. Cowan, *Reliable Computation in the Presence of Noise*, MIT Press, 1963.
6. J.A. Abraham and D.P. Siewiorek, "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," *IEEE Trans. Computers*, vol. 23, no. 7, July 1974, pp. 682-692.
7. R. Sarpeshkar, T. Delbrück, and C.A. Mead, "White Noise in MOS Transistors and Resistors," *IEEE Circuits and Devices*, vol. 9, no. 6, Nov. 1993, pp. 23-29.
8. S.K. Reinhardt and S.S. Mukherjee, "Transient Fault Detection via Simultaneous Multithreading," *Proc. 27th Int'l Symp. Computer Architecture (ISCA 00)*, IEEE Press, 2000, pp. 25-36.
9. J.C. Smolens et al., "Efficient Resource Sharing in Concurrent Error Detecting Superscalar Microarchitectures," *Proc. 37th Int'l Symp. Microarchitecture (Micro 37)*, ACM Press, 2004, pp. 257-268.
10. C. Weaver and T. Austin, "A Fault Tolerant Approach to Microprocessor Design," *Proc. Int'l Conf. Dependable Systems and Networks (DSN 01)*, IEEE Press, 2001, pp. 411-420.
11. S.S. Mukherjee et al., "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," *Proc. 36th Int'l Symp. Microarchitecture (Micro 36)*, ACM Press, 2003, pp. 29-42.
12. N.J. Wang et al., "Characterizing the Effects of Transient Faults on a High-Performance Processor Pipeline," *Proc. Int'l Conf. Dependable Systems and Networks (DSN 04)*, IEEE CS Press, 2004, pp. 61-70.
13. A. DeHon, P. Lincoln, and J.E. Savage, "Stochastic Assembly of Sublithographic Nanoscale Interfaces," *IEEE Trans. Nanotechnology*, vol. 2, no. 3, Sept. 2003, pp. 165-174.
14. J. Han and P. Jonker, "A System Architecture Solution for Unreliable Nanoelectronic Devices," *IEEE Trans. Nanotechnology*, vol. 1, no. 4, Dec. 2002, pp. 201-208.



Darshan D. Thaker is a graduate student in the Department of Computer Science at the University of California, Davis. His research interests include next-generation computer architectures and emerging technologies. Thaker has an MS in computer engineering from Wayne State University.



Francois Impens was a graduate student in the Media Laboratory of the Massachusetts Institute of Technology when he did this work. His research interests include quantum computing and fault tolerance in emerging technologies. Impens has an MS in media arts and sciences from the Massachusetts Institute of Technology.



Isaac L. Chuang is an associate professor of physics and media arts and sciences at the Massachusetts Institute of Technology. His research interests include building large-scale quantum computers and cryptographic systems. Chuang has a PhD in electrical engineering from Stanford University.



Rajeevan Amirtharajah is an assistant professor in the Department of Electrical and Computer Engineering at the University of California, Davis. His research interests include low-power circuit and interconnect design, energy scavenging, and DSP. Amirtharajah has a PhD in electrical engineering from the Massachusetts Institute of Technology.



Frederic T. Chong is a professor of computer science at the University of California, Santa Barbara. He was with the University of California, Davis, when he did this work. His research interests include novel computing technologies, secure and reliable systems, and next-generation embedded and sensor architectures. Chong has a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology.

■ Direct questions and comments about this article to Darshan D. Thaker, Univ. of California at Davis, 2063 Kemper Hall, 1 Shields Avenue, Davis, CA 95616; ddthaker@ucdavis.edu.