Appearing in *IEEE International Workshop on Design and Test of Nano Devices, Circuits, and Systems*, Sept. 29–30, 2008

The Case for Reconfigurable Components with Logic Scrubbing: Regular Hygiene Keeps Logic FIT (low)

André DeHon
Department of Electrical and Systems Engineering
University of Pennsylvania
200 S. 33rd St., Philadelphia, PA 19104
andre@computer.org

Abstract

As we approach atomic-scale logic, we must accommodate an increased rate of manufacturing defects, transient upsets, and in-field persistent failures. High defect rates demand reconfiguration to avoid defective components, and transient upsets demand online error detection to catch failures. Combining these techniques we can detect in-field persistent failures when they occur and reconfigure around them. However, since failures may be logically masked *for long periods of time, persistent failures may accumulate silently; this integration of errors over time means the effective failure rate for persistent errors can exceed transient upset rates. As a result, logic* scrubbing *is necessary to prevent the silent accumulation of an undetectable number of persistent errors. We provide simple analysis to illustrate quantitatively how this phenomena can be a concern.*

1. Introduction

Continued feature scaling means our devices and wires are made of fewer atoms, increasing their fragility. We are approaching an *inflection point* where we can no longer overstress devices to force weak devices to fail before the component is integrated into an end system. *This means many weak devices will turn into defects during operation* [2]. Borkar suggests that by 2016 it may not be unreasonable to anticipate that 10% of the transistors in a component will become unusable during its operational lifetime [1].

Continued scaling also means higher defect rates and transient upset rates. High device defect rates (*e.g.* 1–20%) mean we cannot expect the billions of non-memory devices on a component to all yield perfectly, demanding some form of reconfiguration (*e.g.* [5, 3]). High transient upset rates demand continuous self-checking (*e.g.* concurrent-error detection) and recovery (*e.g.* rollback and retry, voting).

When a device has a persistent failure during the operational lifetime, we call it a *lifetime* failure to distinguish it from transient upsets. Continuous self-checking will detect lifetime failures. If a rollback-and-retry recovery works, then the system knows the failure was transient; when it fails, the system can identify the failure as a lifetime failure rather than a transient. With spare capacity, the systems can reconfigure to avoid the newly failed device.

This strategy suggests that the mechanisms necessary to mitigate against transient upsets and manufacturing defects may already be sufficient to mitigate against lifetime failures. The finite number of devices on a chip imply that viable lifetime failure rates are bounded. Using Borkar's example, if 10% of the 100 Billion transistors on a device fail over a 10 year lifetime, this implies one new transistor defect every $3\times10^8\text{s}/\left(10^{11}\times0.10\right)\approx30\,\text{ms}$. Operating at 10GHz the probability that a logic device fails in a given cycle is $0.1/\left(3\times10^8\text{s}\times10^{10}\text{cycles/s}\right)=3.3\times10^{-20}$.

When the transient upset rate is above the lifetime fault manifestation rate, we might conclude that we do not need any further mechanisms to protect against undetected errors due to lifetime faults. However, transient upsets and lifetime faults are different in an important way. *Logical masking* of errors serves to reduce the effective transient upset rate, while it allows lifetime faults to accumulate over time (Section 3). This demands a different analysis (Section 4) to assess the impact of lifetime faults and may demand additional mitigation mechanisms (*e.g.* scrubbing (Section 6)) to avoid having silent data corruption (SDC) from lifetime faults drive the overall device reliability (FIT rate).

The goals and contributions of this paper are to:

- Articulate how lifetime faults differ from transients with respect to continuous online detection.
- Provide simple, first-order analytical calculations to capture the effects of lifetime faults.
- Highlight scenarios where it is necessary to guarantee frequent path sensitization to maintain high reliability.

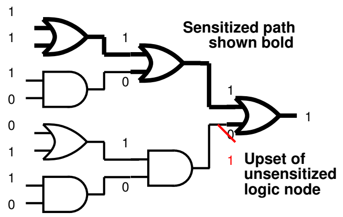

Figure 1. Node upsets that are not sensitized by the current input vector are masked and do not propagate to the observable outputs.

We show cases where the per device per cycle lifetime failure rate is below the transient upset rate, but the FIT rate due to lifetime failures is orders of magnitude higher than the FIT rate due to transient upsets.

2. Background

Related Work Scherrer and Steininger addressed the issue of "dormant" faults at a higher level of abstraction in [10], similarly establishing the high impact that infrequently exercised logic can have on system reliability and the need for online test. The analysis here is performed at the gate level in the context of fine-grained reconfigurability, and the models relate FIT rate to system size.

System Context While the phenomenon addressed by this paper is very general, we consider a specific system model to simplify discussion and concretely illustrate the concepts. We assume a fine-grained reconfigurable substrate such as the nanoPLA [4] which can easily deal with individual crosspoint and wire defects in the 10% range using matching [5]. Logic is decomposed into small blocks, duplicated, and compared at the bit level to detect transient upsets or lifetime failures [8]; streaming buffers separate computational blocks and facilitate rollback and retry once an error has been detected. A watchdog circuit counts rollbacks and invokes diagnosis and reconfiguration repair when too many rollbacks occur. Alternate detection mechanisms (*e.g.* end-to-end integrity checks, arithmetic self checking) can be less expensive than duplication and will be preferable for particular design implementations, but each requires separate detailed analysis which is beyond the scope of this short paper.

3. Logic Masking

An upset along a circuit path which is not *sensitized* by the current set of inputs (See Fig. 1) does not propagate to the output and is not *observable*. We say such upsets are *logically masked* by the input vector. Even if a transient upset propagates to an output or latch, it may not impact the outcome of a computation—*e.g.* errors in branch-prediction logic do not impact the results computed by a processor.

Computing the upset rate without accounting for masking is highly conservative. *E.g.*, if the sensitized path is only 10% of the circuit, the observable failure rate is one tenth the raw gate upset rate. Coupled with bit errors that do not actually impact the computation, this has led to the development of an Architecture Vulnerability Factor (AVF) for processors [7]. AVFs can be below 10% (*e.g.* [11]).

If the failure is a lifetime fault rather than a transient, the gate continues to compute erroneous values after the initial manifestation. In the simplest cases, the lifetime fault may appear as a new stuck-at fault. If a path effected by the fault is soon sensitized by a subsequent input vector, the error is observable and can trigger correction. However, if the lifetime fault occurs on an infrequently sensitized path, the fault may stay around unnoticed in the circuit for a long period of time. During this period of time, the circuit may accumulate other non-sensitized, persistent failures. Of concern are the accumulation of a complementary set of persistent errors that prevent detection. For example, in the duplication scheme, two gates serving the same role in each of the replicas may fail during a long period of non-sensitization; when they are subsequently sensitized the comparison fails to detect the fault since both copies report the same erroneous value. Duplication may be adequate for transients because the probability of such a simultaneous failure on a single cycle is very small; however, when we integrate failures over a large number of cycles, the probability that our observation occurs at a time when complementary errors occur is much higher.

4. Analysis and Quantification

We start by looking at a single gate and its duplicate. If these two gates fail simultaneously, then we have an undetected error (SDC). Starting with the transient case, we assume a raw, gate-level transient upset rate of P_{tu} and a gate sensitization probability of P_s. This gate pair contributes an undetected error when it is sensitized and both replicas fail:

$$P_{transient\_sdc} = P_s \times (P_{tu})^2 \tag{1}$$

In the lifetime case, we assume a raw, gate-level lifetime fault rate P_{lf}. The gate pair fails if the path is sensitized and both gates either fail on this cycle or have failed since the last sensitization event:

$$P_{lt\_sdc} = P_s \times (P_{lf} + P_{f\_unsensitized})^2 \tag{2}$$

Eq. 2 requires we compute the probability that a gate fails since it was last sensitized. That occurs if the gate failed on any of the previous non-sensitized cycles, and can be computed by summing up the probability that the gate was not sensitized for at least m cycles and failed m cycles ago.

$$P_{f\_unsensitized} = \sum_{m=1}^{\infty} \left((1 - P_s)^m P_{lf} \left(1 - P_{lf}\right)^{(m-1)} \right)$$

Assuming P_{lf} very small (*e.g.* $P_{lf} < 10^{-19}$ noted in the introduction), $(1 - P_{lf}) \approx 1$ and the equation reduces to:

$$P_{f\_unsensitized} = P_{lf} \times \sum_{m=1}^{\infty} \left((1 - P_s)^m \right) \qquad (3)$$

Since $\sum_{m=0} (a_0 \times r^m) = \left(\frac{a_0}{1-r} \right)$:

$$P_{f\_unsensitized} = P_{lf} \times \left(\frac{(1 - P_s)}{1 - (1 - P_s)} \right) = P_{lf} \left(\frac{1}{P_s} - 1 \right)$$

Substituting this into Eq. 2, we get:

$$P_{lt\_sdc} = P_s \times (P_{lf})^2 \times \left(1 + \frac{1}{P_s} - 1 \right)^2 = \frac{(P_{lf})^2}{P_s} \quad (4)$$

Comparing Eqs. 4 and 1, this supports the qualitative argument that logical masking effects these two cases in opposite ways. If our gate is sensitized 10% of the time ($P_s = 0.1$), then the masking effect reduces the probability of an undetected transient error by 10× while increasing the probability of an undetected lifetime failure by 10×. If $P_{lf} = P_{tu}$, the FIT rate due to lifetime failures would be $(P_s)^{-2}$ higher; that is, 100× when $P_s = 0.1$.

Across an entire chip, the chip has an undetected error if any of the observable output bits has an undetected error. Roughly, we can sum up the FIT rate for the observable bits. There are several technicalities this raises:

1. The sensitization probability differs from gate to gate.
2. The gates are sensitized in correlated ways.
3. In addition to joint failure of the exactly corresponding gates in the replicas, undetected failures can arise from (i) complementary failures along the sensitized paths which drive the observable output to the same value or (ii) an error in one replica and the comparison logic.

The first demands that we work with a per gate sensitization probability. All three mean that the detailed netlist logical structure will be necessary to perform an accurate calculation. Without examining logic structure, we can compute upper and lower bounds on the FIT rate.

We compute an optimistic lower bound assuming that the paired gate failures in the previous section dominate failure:

$$FIT_{lt\_chip} = \sum_{\text{all gates } g} \left(FIT_{lt_g} \right) \qquad (5)$$

Here we compute the gate FIT, FIT_{lt_g}:

$$FIT_{lt_g} \quad = \quad 10^9\text{hrs} \times 3600\text{s/hr} \times \frac{1\text{s}}{T_{cycle}} \times P_{lt\_sdc}$$

To compute a conservative upper bound, we might assume that **any** second gate failing in the sensitized cone of the replica logic **or** the comparison logic would lead to a failure. Roughly, this means that instead of Eq. 2 we have:

$$P_{lt\_sdc} = P_s \times \frac{(n + c)}{2} \left(P_{lf} + P_{f\_unsensitized} \right)^2$$

Here n is the number of gates in the sensitized cone and c is the number of gates in the checker. The factor of 2 avoids double counting gates. In [8], n is 10–100 and $c < 10$, so the conservative upper bound will be an order of magnitude or two higher than the optimistic lower bound above.

Finally, we quantify the implications for the 2016 component with 10 Billion logic transistors and a 10GHz clock. Starting with the case of homogeneous sensitization rates.

$$FIT_{lt\_chip} \quad = \quad 10^{10} \times 3.6 \times 10^{12}\text{s} \times 10^{10}\text{/s} \times \frac{(P_{lf})^2}{P_s}$$

Assuming that 10% of the transistors will fail over a 10 year operational lifetime, we have $P_{lf} = 3.3 \times 10^{-20}$.

$$FIT_{lt\_chip} = 3.6 \times 10^{32} \times 1.1 \times 10^{-39} \left(\frac{1}{P_s} \right) = \frac{4 \times 10^{-7}}{P_s}$$

This provides us one FIT or lower as long as $P_s > 4 \times 10^{-7}$ based on our optimistic lower bound; this might be $P_s > 10^{-5}$ with a more conservative or precise calculation. For 10GHz operation, $P_s > 4 \times 10^{-7}$ means every device must be exercised at least once every quarter of a millisecond.

However, all we need is a few devices with lower sensitization rates to increase the FIT rate substantially. If we have only 10,000 devices which where were sensitized only once a day, we would have $FIT_{lt\_chip} \geq 3600$.

Since path sensitization is data dependent, it is difficult to reason about how small P_s can be or guarantee it will have any bound. Since P_s can be arbitrarily small, the FIT rates for lifetime failures can be arbitrarily worse than transient upsets even for the same $P_{tu} = P_{lf}$.

5. Infrequently Sensitized Checkers

In our replicate-and-check scenario, the failure path for the checkers is infrequently exercised such that it can drive up the component FIT rate. This provides a concrete example which demonstrates small P_s's are not implausible.

In the presence of lifetime failures, we must, at least, duplicate the comparison logic to avoid a single point of failure. If the checker fails such that it is always signaling a valid match between the replicas, it will not correctly detect when a mismatch occurs. In the transient case, the only single-fault case for the checker was a false positive—the checker signals a mismatch when there is no mismatch.

Nonetheless, even with the dual checkers, the mismatch path is infrequently activated. If the fanin cone for the logic in each replica being checked contains n gates, then the probability of a mismatch is no larger than:

$$P_{s_{mismatch}} = P_s \times 2n \times P_{tu} \quad (6)$$

That is, we get a mismatch when one of the sensitize gates in either replica is upset. Using $n = 100$ and assuming $P_s = 1$ as the best case scenario representing the most frequent exercise of the mismatch case and considering $P_{tu} = 10^{-18}$, a transient error rate which can be tolerated using duplication and checking [8], we get $P_{s_{mismatch}} = 2 \times 10^{-16}$. The checkers for this scenario are likely to make up 1–10% of the logic. Here we assume 1% to represent a minimal case. For a 10 Billion transistor logic design, this means 100 Million transistors are in the checkers.

$$FIT_{lt\_chk} = 10^8 \times 3.6 \times 10^{22} \times 1.1 \times 10^{-39} \times \frac{10^{16}}{2} = 2 \times 10^7$$

This is, of course, an unacceptably large FIT rate. From this example we can see that even if we assumed the checkers composed 2–3 orders of magnitude fewer components or that blocks were 2–3 orders of magnitude larger, we would still have an unacceptably large FIT rate.

Good system design attempts to make the error and exception cases uncommon. As a result, error cases naturally have low sensitization (P_s). This checker example illustrates that if we do not make sure they are exercised frequently, when lifetime faults are probable, these infrequent error paths could drive unacceptable component FIT rates.

6. Scrubbing

Guaranteeing that all paths are sensitized at some minimum rate can be a challenging requirement to place on a design and to verify, especially when path sensitization is data dependent. The straightforward solution is to design in a periodic hygiene routine to guarantee that all paths are sensitized regularly—that is, some form of periodic test or scrubbing of the logic. Logic scrubbing is needed for exactly the same reason we need to scrub memory words in DRAM [9]—since data access is data depenedent, we cannot guarantee that simply correcting data when it is accessed will be sufficient to prevent the accumulation of an uncorrectable (undetectable) number of errors. Many techniques have already been proposed for online, periodic self test (*e.g.* [6]), and this analysis underscores why such testing is valuable. Nonetheless, efficiently sensitizing 10 Billion logic transistor designs every 0.25 ms will likely demand more extreme techniques than previously proposed.

7. Conclusions

Logic masking effects lifetime failures and transient upsets in a different way. Lifetime failure protection can share some mechanism with transient protection, but the different impact of masking requires separate treatment. Particularly, a mechanism is needed to guarantee frequent sensitization to prevent the accumulation of errors which may result in silent data corruption. Logic scrubbing is the most general-purpose solution to mitigate this potential problem.

8. Acknowledgements

This research was funded by National Science Foundation grants CCF-0403674 and CCF-0726602. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Borkar. Microarchitecture and design challenges for gigascale integration. <http://www.microarch.org/micro37/presentations/MICRO37%20Sborkar.pdf>, December 2004. Keynote talk *Int. Symp. on Microarchitecture*.

[2] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, November–December 2005.

[3] W. B. Culbertson, R. Amerson, R. Carter, P. Kuekes, and G. Snider. Defect tolerance on the TERAMAC custom computer. In *FCCM*, pages 116–123, April 1997.

[4] A. DeHon. Nanowire-Based Programmable Architectures. *ACM J. Emerg. Technol. Comput. Syst.*, 1(2):109–162, 2005.

[5] A. DeHon and H. Naeimi. Seven Strategies for Tolerating Highly Defective Fabrication. *IEEE Des. Test. Comput.*, 22(4):306–315, July–August 2005.

[6] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici. Dynamic fault tolerance in FPGAs via partial reconfiguration. In *FCCM*, pages 165–174, 2000.

[7] S. S. Mukherjee, C. T. Weaver, J. Emer, S. K. Reinhardt, and T. Austin. Measuring architectural vulnerability factors. *IEEE Micro*, 23(6):70–75, November–December 2003.

[8] H. Naeimi and A. DeHon. Fault-Tolerant Sub-lithographic Design with Rollback Recovery. *Nanotechnology*, 19(11), March 19 2008.

[9] A. Saleh, J. Serrano, and J. Patel. Reliability of scrubbing recovery-techniques for memory systems. *IEEE Trans. Rel.*, 39(1):114–122, 1990.

[10] C. Scherrer and A. Steininger. Dealing with dormant faults in an embedded fault-tolerant computer system. *IEEE Trans. Rel.*, 52(4):512–522, December 2003.

[11] N. J. Want, J. Quek, T. M. Rafacz, and S. J. Patel. Characterizing the effects of transient faults on a high-performance processor pipeline. In *Proc. Intl. Conf. Dependable Sys. and Nets*, pages 61–70, 2004.