Silent Data Errors: Sources, Detection, and Modeling

Adit Singh[*] Sreejit Chakravarty[§] George Papadimitriou[†] Dimitris Gizopoulos[†]
[*]*Auburn University,* Auburn, AL, USA, singhad@auburn.edu
[§]*Intel,* Santa Clara, CA, USA, sreejit.chakravary@intel.com
[†]*University of Athens,* Greece, {georgepap | dgizop}@di.uoa.gr

***Abstract*—Chip manufacturers and hyperscalers are becoming increasingly aware of the problem posed by Silent Data Errors (SDE) and are taking steps to address it. Major computing facilities operators like Meta and Google have emphasized the critical role of SDEs in today's microprocessors. Numerous studies in the literature have highlighted the severity of this issue, especially in datacenter applications operating at large scales. These errors can lead to data loss and require a significant amount of time and effort to resolve through debugging engineering efforts, which can take months to complete. In this paper, we provide an overview of the issue of SDEs, including an explanation of the problem and the current methods used to address it, as well as gaps that still exist in addressing the issue. We also discuss the different sources of SDEs, including post-manufacturing testing failures, voltage and timing marginalities, and hard-to-detect faults. The paper emphasizes the impact of timing marginalities as a significant source of SDEs. Finally, our spotlight points to the architecture and system dimensions of the problem: we describe the challenges of measuring the true (still unknown) rates of SDE from CPUs, and emphasize on the role of detailed microarchitectural simulation models for this purpose. We present data on the severity of SDEs and their predicted rates under various operating conditions, sources of faults, and technology fabrication nodes.**

***Index Terms*—Silent data corruptions, microprocessors, timing marginalities, voltage failures, microarchitectural simulation, microarchitectural modeling, fault injection, failure rates**

I. Introduction

In modern computing systems, microprocessors are critical components that power a wide range of applications, from personal devices to large-scale data centers. Extreme performance levels are expected to be reached by future supercomputers by leveraging millions of microprocessor cores and specialized accelerators. Ensuring dependability is one of the most difficult barriers to overcome in achieving exascale computing [1]. Microprocessors employ the most aggressive design and manufacturing techniques for the above purposes and are, therefore, far from immune to errors, and in particular, the occurrence of silent data errors (SDEs) can have serious consequences for system reliability and computational accuracy [2]–[4]. Silent data errors are one of the most insidious and difficult-to-detect problems in modern computing. A silent data error occurs when data is corrupted in such a way that it still appears valid (no hardware or software level ”alarm” is raised), but produces incorrect results when used in computation. In the context of microprocessors, these errors can arise from a variety of sources, including cosmic rays, hardware defects, and hardware bugs. One of the known sources of silent data errors is cosmic rays. These high-energy particles can strike computer memory and cause bit flips, where a 0 becomes a 1 or vice versa. Other sources of silent data errors include hardware defects due to manufacturing and aging, hardware design bugs, and power supply fluctuations. Further, low voltage operation makes the chip more susceptible to silent data errors, since the critical charge to flip a bit is lower at reduced voltages [5]–[12].

Identifying and measuring silent data errors in microprocessors can be challenging because they often occur sporadically and are difficult to reproduce [13], [14]. To minimize the impact of on-chip memory errors, error correcting codes (ECC) are used to detect and correct such errors [15]. However, the use of ECC methods results in additional storage requirements and increased complexity, and cannot detect or correct all hardware-induced errors [16]. Although commonly used ECC methods can identify and correct some faults, they are limited in their ability to do so, with the most common method, single error correction, double error detection (SECDED), able to detect up to two flipped bits and correct only one flipped bit per 64 bits [15], [16]. In addition, multiple-bit faults are more common in on-chip memory structures in newer fabrication technologies [17]. While ECC can be beneficial in reducing failure rates in some on-chip memory structures, it is not always applicable to all functional, control, and memory blocks of the microprocessor. Even when using ECC methods, silent data corruptions are still possible, especially in large-scale datacenter infrastructures, which poses a significant threat to program integrity [13], [14], [18].

Silent data errors constitute a significant challenge for modern microprocessors and the computing systems they power. However, through the use of sophisticated error detection techniques, as well as the development of fault-tolerant and error-correcting models, researchers have made significant progress over the last decades in mitigating the rate and impact of silent data errors. As computing systems continue to become more complex and more critical to our daily lives, it is likely that this area of research will continue to be an important focus for the computing community. It is therefore critical to identify the sources of errors that are most likely to affect the program's execution silently, to propose novel ways for modeling and detection of silent data errors. One approach is to use fault tolerance techniques, such as redundancy or replication, to ensure that multiple copies of critical data are available. This can help ensure that even if a silent data error occurs, the system can continue to operate correctly. Another approach is to use error-correcting codes that can detect and correct errors automatically. These techniques can be especially important in safety-critical systems, where the consequences of a silent data error can be catastrophic.

In this paper, we summarize the importance of SDEs by first defining the SDE problem. We present the current approaches and the vital gaps in addressing the important problem of SDEs, and the potential sources of failure that could cause SDEs. We also discuss the impact of hard-to-detect faults that escape from post-manufacturing testing on SDEs, by investigating several types of manufacturing test, such as cell aware tests, scan timing tests, and system level tests. We stress another major source of potential SDEs, which is the sensitivity to voltage failures and timing marginalities, and discuss that timing marginalities being the source of a significant number of SDEs. Finally, we present the challenges in measuring the SDE rates on real microprocessors and in detailed low-level simulation models, and present the severity of SDE rates in different technology fabrication nodes and under several operating conditions using early microarchitecture level modeling and measurement of silent data errors.

II. Silent Data Error Problem

Sreejit Chakravarty

After defining the SDE problem, the methodology used to address it is summarized. and gaps in the current approach are highlighted. Silicon error sources are shown in Fig. 1(b). Fig. 1(a) abstracts the silicon life cycle and depicts how silicon error sources are targeted. Logic bugs can also cause errors but are assumed to be eliminated prior to productization of the product and not considered. Silicon provider's high-volume manufacturing (HVM) flow consists of two screens to weed out faulty silicon: ATE tests and system level tests (SLTs). ATE tests are primarily structural tests like scan, memory BIST, etc. SLTs are application-based tests. Both screens target defects, circuit marginality issues and early life failures. HVM manufacturing screens are not perfect and there are residual faulty silicon escapes, referred to as HVM DPM escapes.

After shipment, silicon is assembled into the final system and undergo additional tests. These tests are more exhaustive system tests, than those used by silicon providers. The goal is to screen for HVM DPM escapes.

A. SDE Definition

Silent Data Error is often discussed in two different contexts. It is important to distinguish between the two since the required solutions are different.

1) **Low DPM SDE.** The first of this is a new name for "DPM escape". DPM escape from HVM testing is a fact of life. But, as shown in Fig. 2(a), it has a different impact on the end customer based on the size of the installation base. DPM1, which could be in the low 100s, is an adequate DPM level for installation base that are small, may be 10s of thousands. However, DPM1 is not acceptable for larger installation base since it leads to unacceptable levels of failure. For larger installation bases a much lower DPM level, DPM2 which could be in the low 10s is required. Achieving such low DPM level is difficult by itself but has been exacerbated due to the rise in the complexity of silicon designs. Fig. 2(b) shows the increase in design size resulting from complex designs incorporating added functionality on a single piece of silicon. This has prompted companies having large installation base to highlight the issue [13], [14]. The central problem, as pointed out in these papers, is the root causing the failures and finding an effective fix to plug the small DPM escapes from silicon providers.
2) **InField SDE.** In Fig. 1(a), InField refers to the silicon life cycle phase when the end user uses the silicon device. InField errors are radiation induced soft-errors or aging induced reliability hard failures. Protection mechanisms, like memory error-correcting codes (ECC), are added for InField errors. InField errors are said to be silent if the protection mechanism does not detect them. For example, an error in the memory that is not detected and/or corrected by the ECC scheme is a silent error. It

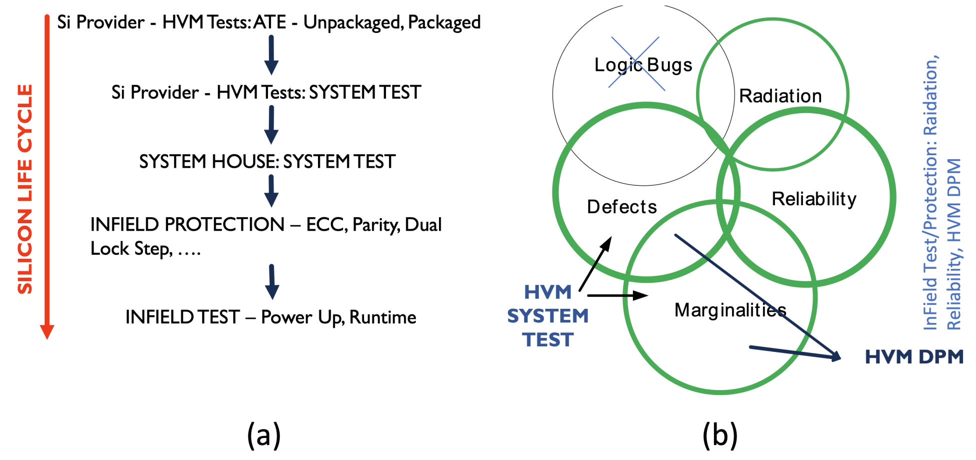
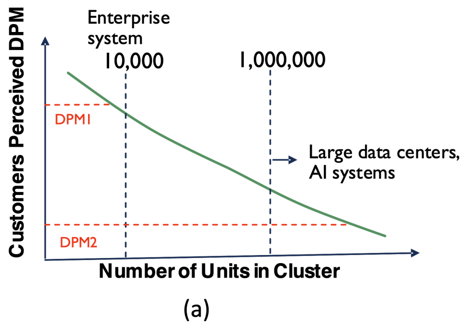

SILICON LIFE CYCLE
Si Provider - HVM Tests: ATE - Unpackaged, Packaged
Si Provider - HVM Tests: SYSTEM TEST
SYSTEM HOUSE: SYSTEM TEST
INFIELD PROTECTION – ECC, Parity, Dual Lock Step, ….
INFIELD TEST – Power Up, Runtime
(a)
Logic Bugs
Radiation
Defects
Reliability
Marginalities
HVM SYSTEM TEST
HVM DPM
InField Test/Protection: Raidation, Reliability, HVM DPM
(b)

Fig. 1. Mapping of silicon error sources to silicon life cycle.



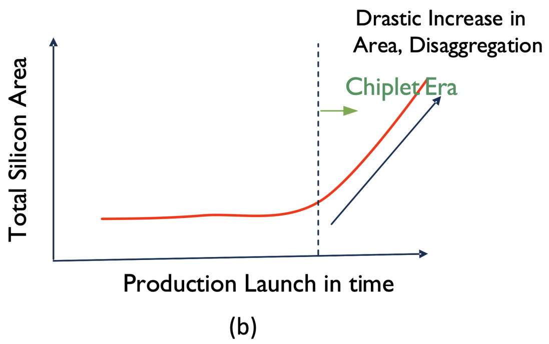


Fig. 2. DPM impact on customer installation.

is silent in the sense that the silicon does not recognize or flag the error. The resulting error could be identified in other system components, like firmware, or not.

B. Current Approach and Gaps in Addressing SDE

1) Brick and Mortar Story of Low DPM SDE: Bick-and-mortar analogy is used to address the current methodology for Low DPM SDE problem. For ATE structural testing, the SoC is divided into small DFT domains like memories, logic within a clock domain, analog IPs, etc. Specialized DFT added to each DFT domain, shown as bricks in Fig. 3(a), is used to test it. Inter-domain structural tests, like hierarchical scan, are used to test the silicon interface between DFT domains, shown as mortars in Fig. 3(a). However, inadequate coverage or inadequacy of the target fault model result in DPM leaks, shown as brick test drips in Fig. 3(b). Inadequacy of inter-domain structural tests results in mortar drips of Fig. 3(b). "Mortar test sealants" (HVM-SLT) of Fig. 3(c), which are HVM SLTs, are added to patch these leaks. There are two major issues with this HVM test flow.

- Structural tests are based on fault models, and work well, unless very low outgoing DPM is a requirement. Ongoing effort to improve fault models, to reduce HVM DPM, targets only the "brick test drips" of Fig. 3(b).
- Filling "mortar drips" of Fig. 3(b) is getting more difficult due to the increase in design size and aggressive design trends. The inadequacy of fault model based structural tests and automation tool capacity are falling short in addressing this problem.
- HVM-SLTs, which are randomly generated tests with different instruction and data mixes, use a "shot gun" approach to fill the brick-and-mortar drips. Increasing the number of HVM-SLTs improves its quality. However, there is no meaningful approach to identify effective HVM-SLTs or to measure an HVM-SLT's quality. This approach adds very significantly to test cost. They are therefore truncated, leaving a DPM gap.
- As shown in Fig. 3(d), system houses al run a barrage of tests before enabling their systems for end user use. These are a mix of frequently used applications, mixed in with some longer tests from the HVM system test flow. Such tests run for longer duration, and based on the usage model, run often enough prior to farming it out to the end user. Specific tests that fail are fed back to the silicon provider to plug their test holes.

Hence, we have reached an inflection point and new ideas are needed to fill the gap left by the adhoc HVM test screens.

2) InField SDE: InField error prevention relies on protection mechanisms which are assumed to provide good coverage. This is a fallacious assumption! Protection schemes were developed to protect against radiation induced soft errors and provide poor protection against aging and HVM DPM related hard failures, as illustrated using the example of Fig. 4. Fig. 4(a) shows a memory with data ECC and address parity protection. Fig. 4(b) shows a detailed view of the array. Note that sense amplifiers (SA) do not change state if its differential inputs are not driven.

Assume wordline W0 s@0. On a read from location 0, the cells are not selected, the SA's differential inputs are not driven, and memory output retains the old state. If the previous read was fault-free, then the ECC decoder will not detect this error. For the memory writes and reads sequence of Table I, the output after the second read should be BB. In the faulty case, it is AA instead. However, since the ECC[AA] matches

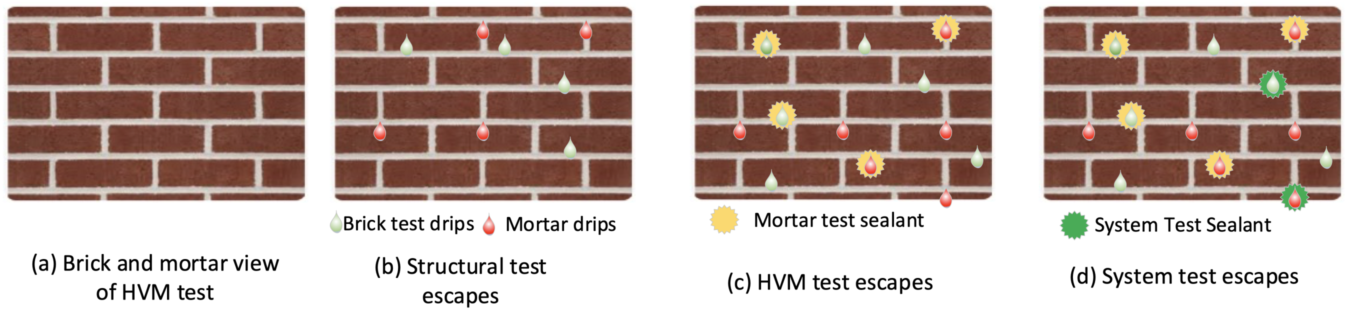


Fig. 3. Brick-and-Mortar analogy of Low DPM SDE issue.

TABLE I
A PERSPECTIVE ON SDE SOURCES.

Memory Operations			Good Data	Faulty Data
1st	Write	ADDR3, AA	N/A	N/A
2nd	Read	ADDR3	AA, ECC[AA]	AA, ECC[AA]
3rd	Write	ADDR0, BB	N/A	N/A
4th	Read	ADDR0	BB, ECC[BB]	AA, ECC[AA]

the faulty data AA the decoder will not flag this error. This is an example of an InField SDE. A similar observation can be made using reliability studies of ECC protected memories alluded to in Fig. 4(c). It shows the percentage of reliability induced errors escaping detection (and correction) by ECC decoders, at different voltage and operating corners.

The key question is: how is the gap left by protection schemes filled? Repeated or periodic InField testing is the predominant practice. Unfortunately, there is a major gap in InField testing.

- Scan/LBIST tests with 90% or more coverage is often considered to be very good logic test.
- March C- is assumed to be a good infield test for memories.

What is the basis for using the above tests for InField testing? It seems to be based on convenience rather than any sound engineering basis. In addition to the above structural tests, SLTs are often used as InField tests. Once again, based on our inability to measure the quality of SLTs there is no rationale basis for the use of such tests. They are more of a feel-good test! Questions that remain unanswered are: what is an appropriate InField test for logic and memories? How often should we apply these tests? How do we create measurably effective SLTs for InField Testing?

III. TESTING TIMING MARGINALITIES THAT CAUSE SDES

Adit Singh

A. *Understanding Escapes from Manufacturing Tests*

The previous section has outlined the many varied sources of failure that can potentially cause silent data errors (SDEs) in computing systems. Any defective, marginal, or unstable IC or SOC that is not detected and screened out during post manufacturing tests can potentially cause failure during operational deployment. additionally, early life failures may occur from latent defects of device aging. While many test escapes result in repeated and severe malfunction during operation, including system crashes, that are readily observed and detected, it is not uncommon for more subtle faults with a limited error impact to go unnoticed, particularly if they are infrequently activated. Observe that rare activation with minimal error impact are precisely the characteristics of the "hard to detect faults" that escape post manufacturing testing in the first place.

1) New Cell Aware Tests: The solution to minimizing test escapes that cause field failures is obviously better testing, with more complete test coverage of the actual defects and anomalies observed in manufacturing. This was discussed at some length in the previous section. Classical stuck-at (SA) and transition delay fault (TDF) test generation explicitly targets faults only at the circuit nodes, i.e., the interconnects between the standard cells; defects and faults within the standard cells of the design are not targeted during test generation, and are only serendipitously detected. Plugging this test coverage gap was the motivation behind the recent introduction of the Cell Aware (CA) test methodology [19] which also targets shorts and opens within the standard cells of the design during test generation. CA tests have been shown to significantly reduce test escapes in volume production. However, comprehensive CA test generation significantly increases test set size and test application times. Additionally, targeting resistive defects, beyond ideal shorts and opens, can make test costs prohibitive. Consequently, CA tests are generated for only a limited set of defect resistances in practice, resulting in many of the test escapes discussed earlier with the help of the Brick-and-Mortar analogy in Fig. 3. Nevertheless, in principle, effective test methodologies are available today to minimize field failures, including silent data errors, that are caused by classical permanent faults. In this area, the current focus of test development in advancing DFT (design-for-test) and ATPG (Automatic Test Pattern Generation) towards better detection of these faults is largely aimed at reducing the cost of test generation and application to facilitate high coverage CA tests.

2) Scan Timing Tests and Process Variations: Meanwhile, scan test methods that target timing failures appear to be less effective in screening failing state-of-the art circuits. Both tradition TDF, as well as two-cycle CA delay tests, only target localized (lumped) gross delays. While these tests can be effec-

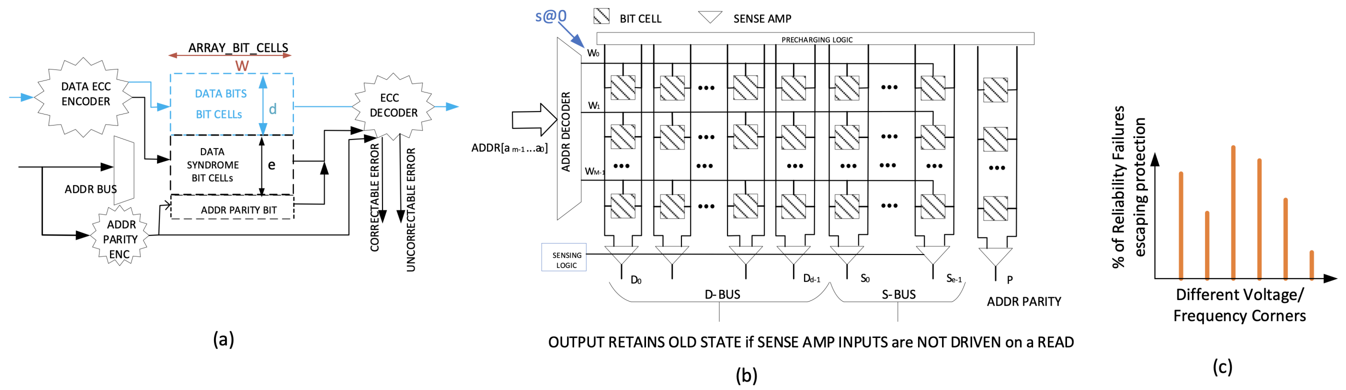


Fig. 4. Silent Data Error of Data and Address protected Memories.

tive in screening out large delay faults caused by mechanisms such as an isolated resistive or open defect, they can miss timing failures caused by an accumulation of distributed delays in a circuit path caused by process variations. Such delay variability is presenting an increasing reliability challenge at advanced technology nodes. Even timing aware scan TDF tests [20], developed for small delay defects (SDDs), have not proven effective against random process variations that can impact every component in a circuit to some degree; each gate or standard cell can display a unique delay. Given the billions of paths in a processor, a significant number (due to pure statistical chance) can contain several gates significantly slower than nominal. Therefore, because the long paths in high performance circuits are carefully optimized during design to all have nearly equal delays to allow fast clock rates, each manufactured instance of a design can have a different slowest path in silicon due to random process variations. Consequently, as has long been recognized, reliable testing to ensure that a circuit meets timing in the presence of significant random gate delay variability requires a comprehensive path delay test. Unfortunately, for many reasons, including the increase in test time due to the very large number of paths that need to be tested, effective scan-based path delay testing [21] has so far not proven practical.

3) System Level Tests: In the absence of an effective scan-based path delay test capability industry, in recent years, is increasing relying on at-speed functional system level tests (SLTs) as a final test screen, primarily to eliminate undetected timing failures. At-speed functional tests have been the gold standard for testing circuit timing in an operation environment, but are expensive both to develop and apply. Also, the test coverage is unknown and can be limited. Therefore, in practice, low-cost scan-based structural tests, applying high coverage test content generated using SA, TDF, and increasingly also Cell Aware (CA) fault models are used during the wafer probe and post packaging test insertions. A comprehensive functional test is then performed at the final stage. This “component” system level test (SLT), that specifically targets only the DUT (Design Under Test), involves temporarily mounting the packaged part on a test board that accurately mimics the intended application hardware, including all its electrical characteristics. The part is then extensively tested in functional operation for as long as an hour or more, at full rated speeds, over a range of user applications and operating conditions. New highly parallelized SLT testers, that allow tight temperature control of the device during test to replicate actual operating conditions, have been developed to support such long test times with high test throughput. However, the coverage of functional tests is difficult to estimate and quantify; test escapes remain a challenge.

B. Voltage Sensitive Failures and Timing Marginalities

1) Industrial Data on Voltage Sensitive V_{min} Failures: The need for aggressive dynamic voltage frequency scaling (DVFS) for thermal management and temperature control in high performance processor SOCs appears to be significantly increasing the occurrence of a class of voltage sensitive timing failures, being referred to as V_{min} failures. These are parts that only fail close to the minimum specified operating voltage, while passing at higher voltages. Table II summarizes some data from a recent industrial test experiment [22] on Intel 14 nm FinFET processors that was aimed at studying the addition fallout observed from CA (cell aware) tests after prior testing with SA and TDF tests. The total additional fallout from CA tests, after screening by the traditional tests, was 4300 DPPM. Out of these, approximately 90% needed two-cycle CA-delay tests for detection. A significant number, 1600 DPPM, were only detected close to V_{min}. While the data clearly highlights the benefits of CA-tests, the large number of V_{min} only failures raise concern about the possibility of additional undetected voltage sensitive marginal parts in the population that may experience failure under less favorable circuit operating conditions. This is illustrated in Fig. 5.

2) Marginal Timing Failures: Fig. 5 plots the longest path delay for each instance of a hypothetical collection of manufactured ICs (Integrated Circuits) of the same design. Worst case path delays for individual instances are different because of manufacturing process variations. Also shown in the figure is the time period and clock edge corresponding to the clock frequency at which the circuit is expected to operate correctly, down to V_{min}. While more tightly clustered together at higher supply voltages, the critical path delays increase and spread out as the supply voltage is lowered towards V_{min}. This is because circuits slow down significantly as supply voltage is lowered. More importantly, the increase in delay of a slow, high threshold voltage, transistor can easily be 3 or 4 times greater close to V_{min}. Low voltage operation greatly accentuates delay variability due to process variations. The plot in Fig. 5 illustrates this spread for a few sample V_{DD} values. At V_{DD} = V_{min}, the spread of worst-case delays in the collection of circuits plotted exceeds the clock period for some parts. These are the V_{min} only failures.

Observe in Fig. 5 that path delays are much smaller than

TABLE II
DISTRIBUTION OF FALL-OUT FROM DIFFERENT CAT TESTS IN [22].

CAT-Static (Before Delay Tests)	**CAT-Delay (Fails at V_{max} & V_{min})**	**CAT-Delay (V_{min} Only Fails)**
400 DPPM	2500 DPPM	1400 DPPM

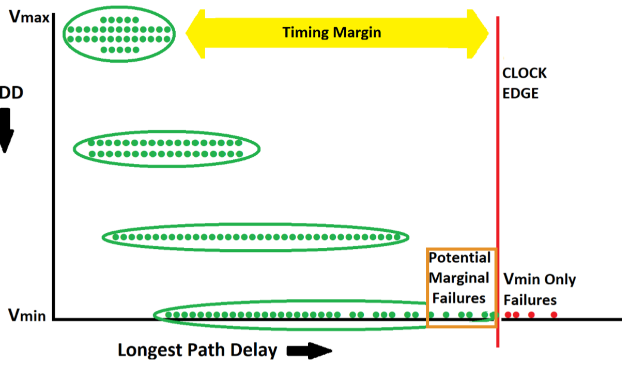


Fig. 5. Critical Path Delays in a Collection of ICs.

the clock period at high values of V_{DD}, allowing a significant noise margin. Thus, even if an occasional IC may fail timing because of an open or short defect, the bulk of the path delays are well clear of the clock edge, and highly unlikely to fail due to circuit noise of other environment conditions. However, at V_{min}, even if all the parts failing the timing test are scrapped, there are many additional marginal parts near the clock edge that narrowly pass timing as shown. These may fail in operation under more unfavorable operating conditions, e.g., power supply noise, that can momentarily lower V_{DD} further. These are marginal parts that may malfunction and cause occasional, unpredictable errors in operation. Unfortunately, as explained below, the specified V_{min} cannot be conservatively raised to increase timing margins and eliminate such failures.

3) Timing Margins Reduced for Power Management: The throughput of high-performance processors has long been limited by the need to manage heat dissipation. Even though clock rates have not increased in two decades to hold down power dissipation, the exponential increase in transistor counts from technology scaling that continues to track Moore's Law means that all the cores in a package cannot always be operated at their maximum frequency. Active thermal management is an integral part of modern processors, where dynamic voltage frequency scaling (DVFS) is used to reduce supply voltage and clock frequency and cool off the die as needed to maintain acceptable operating temperatures. Modern designs offer the on-chip thermal management system a choice of several operating frequencies, each with a specified V_{min}. This is illustrated for a hypothetical processor in Table III. The V_{min} values provide guidance to thermal management on how far the supply voltage can be lowered to save power while operating at each frequency. Observe that since maximum operating voltages today are well below a volt, only modest voltage, and therefore power, reduction is available when reducing operating frequency by each step. For this reason, V_{min} cannot be conservatively chosen; it must be set to be as low as possible. In fact, to maximize power savings at each operating frequency, the V_{min} values are individually estimated and assigned (as shown in Table III) to each processor core. This is to avoid the need to be conservative and use high V_{min} values to accommodate systematic core-to core, and die-to-die parameter variations, if common V_{min} assignments are made to an entire batch of processors. Note, however, that each V_{min} is the best estimate, based on limited test measurements. Since accurate V_{min} measurement involves a search requiring repeating an accurate timing test multiple times at different candidate V_{min} voltages, the cost of obtaining a measured V_{min} value for each core at each target frequency is prohibitive. Unfortunately, since the estimated V_{min} values cannot to overly conservative, this leaves open the possibility of some timing failures at V_{min} for a few outlier parts with delays in the tail of the process variations distribution. These are the parts that fail at V_{min} in Fig. 5. Setting a higher, more conservative V_{min}, e.g., V_{DD} can minimize such failures, but since power consumption is a quadratic function of V_{DD}, this would significantly reduce the power savings for the many parts with near nominal delays. It is implicitly expected that the few parts failing at V_{min} will be screened out by testing.

4) Timing Marginalities and SDEs: Unfortunately, as shown in Fig. 5, when the tail of the worst-case path delay distribution approaches or crosses the clock period boundary, without any timing margin remaining to absorb electrical or environmental noise experienced by the circuit, a significant number of the marginal parts that barely pass the timing test can fail in operation under worse case operating conditions. Any such part that is not detected by subsequent functional systems tests and ends up in a system deployed in the field can exhibit unpredictable and intermittent errors and can potentially be a source of SDEs.

There are hints in the published work that point to timing marginalities being the source of many of such errors. Studies [23] have found that SDE rates are higher in very low frequency, low voltage operation. This is consistent with slow outlier process variability paths being the source of the failures because the delay of such paths is greatly accentuated in low voltage operation. It has also been found the rate of occurrence of SDEs increases with device age. Again, modern transistors and their circuits slow down a little over time due to aging mechanisms such as NBTI. It is, therefore, expected that the delay of marginal paths will increase with time, and the timing error rate will also go up.

C. Detecting Timing Marginalities in Processors

1) Key Characteristic of Slow Outlier Paths: Recent research [24] studying gate and path delay increases from process variations has suggested that because of the extremely large number of transistors in modern SOCs, a noticeable number of devices can be expected to be from the extreme tail of the variability distribution. Published work [25] has found that transistor threshold voltages (V_{th}) are normally distributed, at least out to ± 5 sigma (σ). Assuming that this distribution remains normal, since the probability of a 6σ transistor in about 1 in a billion, every SOC incorporating billions of transistors can be expected to contain many such extreme transistors. A population of a million such SOCs (used to measure DPM) will even contain thousands of 7σ transistors and beyond. Such extreme transistors, with greatly increased threshold voltages, can slow down dramatically when operating at the lowest V_{min} corresponding to the lowest operating frequency in a processor. This is because the transistor gate voltage is not far above V_{th} when the transistor is on, resulting in a very weak gate overdrive. For example, a logic gate containing an extreme 7σ transistors can have 10X or more delay compared to a similar logic gate operating at the same

TABLE III
FREQUENCY AND V_{min} TABLE.

Freq	V_{min}
F1	V_{min_1} = 0.90V
F2	V_{min_2} = 0.79V
F3	V_{min_3} = 0.71V
F4	V_{min_4} = 0.60V
F5	V_{min_5} = 0.49V

low voltage with a nominal transistor. Such outlier transistors can alone contribute to slow outlier paths and significant DPM from timing failures.

The above discussion can be more formally understood based on modeling logic gate delays using an analytical gate delay model, such as the well validated Sakurai-Newton alpha-power law [26]. This approximates switching delay to be proportional to $1/(V_{DD} - V_{th})^{\alpha}$. The literature suggests that for advanced FinFET technologies, the best fit α appears to be 1.25. Observe that the switching delay blows up as $V_{DD} - V_{th}$ approaches zero. At very low power saving operating voltages around 0.5V, the gate overdrive $V_{DD} - V_{th}$ is typically only 150-200mV for nominal V_{th}. Statistically extreme transistors can raise V_{th} sufficiently high to reduce this overdrive to near zero, introducing very large delays in the circuit. Failure analysis has measured functional transistors with V_{th} increase due to variability of more than 200mV.

Given this dramatic non-linear increase in delay caused by the extreme shift in V_{th} in $7 - 8\sigma$ transistors, it has been speculated [24] that many, if not most, of the rare long paths that fail timing due to process variations get most of their increased delay from a single extreme V_{th} transistor. There is credible experimental evidence validating this conjecture. However, given the rare occurrence rates (1 in 10^9 – 10^{12}) of 7σ and 8σ transistors, direct experimental validation appears impractical. Even meaningful Monte Carlo simulations of circuit path delays using SPICE are very time-consuming to perform. Nevertheless, early trends from results of a few hundred million simulations runs appear to confirm that the longest circuit paths due to process variations do contain an extreme outlier transistor. These are the paths that can fail at V_{min}, or are marginal paths with the potential to fail in operation.

2) Screening V_{min} Marginal Parts: Observe that if a target path to be delay tested contains a single extreme outlier transistor that contributes most of the increased delay in the path, then the path exhibits a large lumped delay at the output of the gate containing the slow transistor, which will likely be detected by two-cycle TDF or CA tests. (It is important to remember here that all tests are probabilistic; no test can guarantee that all defects in an IC are detected.) This observation is significant, because it suggests that path delay tests may not be essential to detect timing failures caused by large random process variations. It also explains the unexpected success of CA timing tests in detecting all V_{min} SLT failures in [22].

The two-cycle timing test targeting the extremely weak transistor in the slow path can be further sensitized by conducting the test at V_{DD} 10-25mV below V_{min}. A supply voltage 25mV below V_{min} will further reduce the $(V_{DD} - V_{th})$ term in the Sakurai-Newton equation above, causing an extreme ($> 7\sigma$) or ($> 8\sigma$) transistor in the path to add another 10+ nominal gate delays, significantly slowing down the path further and making it easier to detect. Meanwhile, delays in statistically less extreme transistors (the assumption is that there is only a single extreme transistor in the slow path to be detected) will increase much less. Any risk of incorrectly failing such paths can be mitigated by slowing the clock to increase the clock period by a couple of nominal gate delays to absorb the slowdown in good paths. The optimal supply voltage and clock period/frequency to maximize detection of slow and marginal paths while avoiding yield loss from over-testing will need to be worked out on the test floor from experience, as is common practice.

In summary, research is ongoing to validate and exploit the assumption that many of the slow paths with large delays due to process variations that cause timing failures in advanced processors experience most of their increased delay from a single statistically extreme outlier transistor. This should allow traditional lumped delay scan timing tests to remain effective in detecting such failures. Furthermore, overtesting somewhat below V_{min} with appropriate adjustment in clock frequency can further sensitize the slow path detection during both scan and SLT testing.

IV. The Challenge of Measuring SDE Rates

G. Papadimitriou, D. Gizopoulos

A. The Challenging Task of Unveiling Errors at System-Level

The main challenge of measuring SDE rates is that, per their definition, these errors are silent, which means that no hardware-based or software-based error handling mechanism can detect them. The SDE rates are often low, and strongly depend on the defective hardware structure and the executed software workload. This means that in order to measure SDE rates accurately, a large amount of data must be processed coming from non-negligible numbers of faulty chips. For example, in a typical datacenter, billions of bytes of data may be processed every second. This requires specialized equipment and techniques, such as hardware monitors or software-based profiling tools [27]. Another reason why measuring SDE rates is challenging is that these errors can be highly dependent on the system's configuration and workload. For example, SDE rates may be higher in systems that operate in harsh environments or experience frequent power fluctuations [7], [9], [10]. They may also be higher in systems that run complex and demanding applications. Therefore, in order to accurately measure SDE rates, it is necessary to perform large-scale experiments under a wide range of conditions, which is both time-consuming and expensive; it can practically be realized only by owners of extreme scale systems.

Enterprise and cloud data centers are installing increasingly complex System-on-Chip (SoC) devices in large numbers, which raises the likelihood of undetected faults that can cause unexpected crashes or silent data errors (SDEs). Although soft errors due to cosmic rays are widely recognized [28], [29], the scale of data center infrastructure means that SDEs caused by escaped manufacturing defects and in-field reliability mechanisms must also be taken into account [13], [14], [27]. Defects leading to SDEs are difficult to detect and screen due to the multiple conditions required for their occurrence, such as specific machine instruction sequence, operating voltage, frequency, and temperature conditions, and platform behavior like interrupts [14]. These factors result in limited repeatability of SDE detection tests and the need for extended test duration to identify failures, making it crucial to design test methods with this behavior in mind. One method is to execute SDE-targeting code multiple times during tests, while another is to use pseudo-random instruction and data sequences on every execution loop to increase the number of unique data sequences applied by the tests.

Numerous factors can cause faults in an SoC, such as radiation, electrical marginalities, and manufacturing defects. Even silicon defects that are not detected (or even exist) during manufacturing can result in faults [30], [31] in the field. The way these faults affect the operation of a workload depends on the circuit where the fault occurs [32], [33]. If the fault occurs in a circuit that includes error detection and correction, such as a cache or memory with error correction code, the hardware can correct the error.

Silent data errors go undetected, do not interrupt the machine operation, but instead result in data errors. Data errors are more likely to occur when faults occur in circuits that are not used for program control, such as the SoC's integer of floating-point units [35]. The effects of silent data errors are unpredictable and depend on various factors. While an incorrect calculation of a single pixel value may not be significant, a data error in a financial transaction calculation could require corrective action [27]. Since a single fault can manifest in different ways over time due to workload variations, managing faults that can cause SDE at scale is crucial, particularly when millions of processing cores are installed in a data center or a supercomputer. Lerner *et al.* in [27] presented that a datacenter of modest size (i.e., 100,000 SoCs) is likely to experience at least one SDE event per month with a rate of 10 failures in time (FIT)[1]. For larger installations, frequent SDE events are likely, even at 1 FIT. To this end, it is crucial to minimize the rate of SDE, for example, by periodically testing the datacenter infrastructure to identify defective hardware components that perform wrong calculations.

B. The Need for Billions of Real Machines or Billions of Years of Simulation

Given the challenges associated with measuring SDE rates, discussed in the previous section, it is not surprising that many researchers have turned to simulations as a way to study such errors: simulation-based analysis provides the opportunity to evaluate faulty chips even without having access to any faulty physical chip. Simulations have their limitations. In particular, measuring SDE rates at the RTL (register-transfer level) provides very high detail and accuracy, which, unfortunately, is extremely computationally expensive. In fact, measuring real SDE rates at the RTL is practically impossible since it can take many years, even with the most powerful computers available today. Table IV shows the most common ways to evaluate the reliability (including the expected SDE rates) of computing devices, comparing the time and cost required to complete the study, how many of the available resources can be accessed (or are modeled), if the faults are induced by processes that are natural (i.e., realistic error rates) or synthetic (i.e., models chosen by the user), if the study can be performed in the early stages of the project or only on the final product, and how much information can be gathered on faults generation and propagation (observability). Alternatively, researchers can attempt to measure SDE rates using real machines [13], [14]. However, this approach is possible only for hyperscalers, i.e., owners of huge fleets of computing machines to study SDE rates accurately. For example, in order to precisely measure the SDE rates, billions of machines may be required [13], [14]. Even with the proliferation of cloud computing and big data platforms, it is hard to obtain access to such large numbers of machines. For microprocessors consisting of several million bits and programs consisting of several billions of cycles, determining the real probability of failure (or the FIT rate) is an extremely difficult, if at all possible, task. Specifically, there are two stages at which the FIT rate is measured.

Early stages: when both the design of the microprocessor hardware and the design of the software are under development and major modifications can be applied. In early stages, the FIT rate of a microprocessor, program pair is actually estimated or predicted and not really measured. This is because there are certain parts of the hardware structure of the microprocessor that are still unknown in detail or are deliberately removed from the abstraction to facilitate the design and simulation of the system. Analysis of the failure rates at early stages can be only implemented using architectural (ISA), or microarchitectural models of the system, both of which are available very early in the design flow. Architectural models do not include any hardware information, but only the ISA visible hardware locations (memory and registers) [36]. Microarchitectural models (also referred to as performance models) have a significant detail of the microprocessor hardware: they contain all major hardware storage components that occupy a very large part of the final silicon estate (registers, register files, buffers, queues, caches, predictors, etc.) but they model the combinational logic and the random sequential logic (state machines in control) only functionally. Program exe-

[1] 1 FIT equals to one failure every 10^9 (one billion) hours of operation

TABLE IV
SILENT DATA ERROR RATE MEASUREMENT METHODOLOGIES [34].

Evaluation Method	Time Needed	Cost	Accessible Resources	Fault Source	Availability	Observability
Field, Lifetime data	months/years	very high	all	natural	final product	limited
Beam testing	hours	high	all	natural	final product	limited
Software-level fault injection	hours	low	limited	synthetic	early/final product	medium
Architecture-level fault injection	days	low	limited	synthetic	early	medium
Microarchitecture-level fault injection	**days/weeks**	**low**	**most**	**synthetic**	**early**	**very high**
RTL fault injection	years	low	all	synthetic	late	very high

cutables (assembly/machine instructions) can run on both an architectural and a microarchitectural model. The architectural model is typically around three orders of magnitude faster to simulate than the more detailed microarchitectural one. Moreover, microarchitecture-level models can also be used for bug modeling during the CPU validation phase, e.g., [31], [37].

Late stages: when the microprocessor design, as well as the program design, are very close to completion and design changes (particularly to the hardware) are either impossible or extremely costly. At these stages, the failure rate of a microprocessor, program pair can be actually measured because almost all details of the hardware design are in place, unlike the early models. In late stages, measurement of the failure rates are mainly employed for validation purposes. Late stages measurements can be realized when the program runs on two setups: a gate-level (RT level) model of the microprocessor design, and an actually manufactured silicon chip. Unfortunately, the simulation speed of such fine-grained late stage models is prohibitive to run reasonably long programs. The simulation throughput of the gate level models is typically three or more orders of magnitude slower than the microarchitecture (performance) models. Therefore, the combined effect of the hardware design and software design on the failure rate of the system cannot be measured at the gate or the RTL.

Finally, measurement of the failure rate on actually manufactured chips is the only true physical experiment which runs at the true speed of silicon. The major drawback of this experiment is it requires expensive accelerating testing of the chips with dense beams of particles. Such particles (neutrons or others) are blindly hitting the chip when the program runs, and the failing executions (output corruptions or abnormal terminations) are recorded. There is no way to isolate the hardware spot that was affected and the bits that were flipped. However, the failure rates of such physical beaming experiments are employed by the industry to better emulate the actual physical conditions in an accelerated setup to reach statistically significant results for the failure rates.

Summary: Typically, RAS architects rely either on Statistical Fault Injection (SFI) [38] or on analytical methods, such as the Architecturally Correct Execution (ACE) analysis [39], to provide insights into the programs' resiliency toward transient faults, because both methods aim to report the cross-layer vulnerability. Unlike lower-level simulation models (e.g., gate and RTL), microarchitecture-level fault-injection based on performance models allows deterministic end-to-end execution of large workloads on top of an operating system, i.e., full system analysis, which is impossible at lower levels [33], [36]. Further, injection on RTL models [40] would marginally augment vulnerability analyses with combinational logic vulnerability, since logic has very low raw failure rates compared to storage elements.

We, therefore, employ GeFIN [6], which has been developed and extended on top of the gem5 simulator [41], which is a state-of-the-art microarchitecture-level simulator. Recent studies have shown that fault injection based on microarchitecture-level models in gem5 simulator can provide vulnerability results of the entire CPU during 18 days [33], in contrast to RTL fault injections, which could need several years (see Table IV). In the next subsections, we summarize a number of recent vulnerability studies using our gem5-based simulation and injection set of tools.

C. SDE Failures in Time Analysis

Failures in Time (FIT) rate of a device is the number of failures that can be expected in one billion (10^9) device-hours of operation. For each hardware structure in a microprocessor, a different FIT is computed using the formula below.

$$FIT_{struct} = AVF_{struct} \times rawFIT_{bit} \times \#Bits_{struct}$$

The FIT of the structure is determined by three components: the FIT$_{BIT}$ (or raw FIT) rate, which is determined by the fabrication technology and expresses the fault rate of a single bit, the number of bits of the structure and the SDE AVF of the structure, which is affected by the microarchitecture and the running workload. The raw FIT rate expresses the number of SDEs that will be introduced in the component, while the AVF (architectural vulnerability factor) is the derating factor that quantifies how many of these errors will lead to a failure. The product equals the FIT rate of a component. The SDE FIT rate of the entire CPU is calculated by adding the individual SDE FITs of the individual hardware structures. In the following subsections, the AVF is determined using microarchitecture level fault injection on gem5 simulator.

Fig. 6 shows the SDE FIT rate for each technology node [17]. The red color indicates the percentage of SDE FIT due to multi-bit faults, which starts from 0% in 250 nm node and reaches a high 12% in 22 nm. We can also see that the SDE FIT for each technology node is increasing until the point of 130 nm. After that, the SDE FIT rate starts to decrease, reaching the lowest FIT values at 22 nm. These values correspond to the exact same microarchitecture with the exact same configuration. The differences observed are due to the much smaller area that the chip occupies in the higher density technologies, which results in a significantly smaller number of particles that will eventually strike the microprocessor.

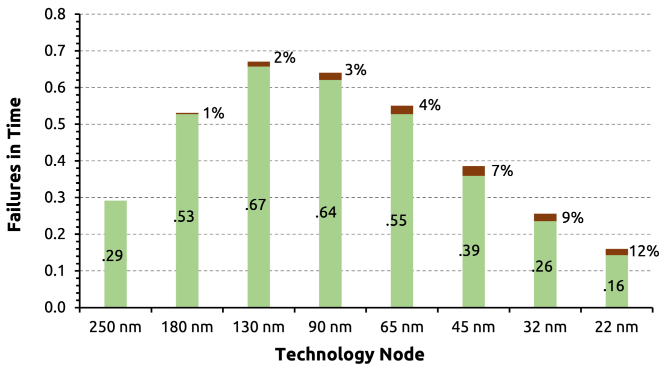


Fig. 6. SDE FIT for the entire CPU core for different technology nodes (numbers inside the green bars) due to transient faults. Red color areas correspond to the contribution of multi-bit upsets. The graph shows only the FIT rate for SDEs [17].

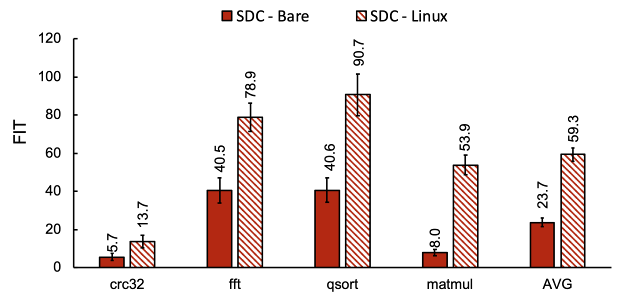


Fig. 7. Cortex A5 Bare-metal and Linux beam FIT rates for SDEs [34].

D. *SDE Rates for Bare-metal versus OS Executions*

In this section, we summarize a characterization study which is performed through physical beam experiments on an Arm Cortex-A5 microprocessor, to show the contribution of OS (Operating System) to the SDE rates. To this end, we show a comparison between SDE FIT rates of bare-metal executions and with Linux OS. In Fig. 7 we can observe that the average SDE rates for A5 is 23.7% for bare metal and 59.3% for Linux. It is also clear from Fig. 7 that the SDE rate is constantly higher when the applications run on top of Linux, in contrast to bare-metal execution. Specifically, we can see that the difference of the SDE rates between bare-metal and Linux OS can be as high as 6.7×. However, as we discussed earlier, during beam experiments it is very difficult to investigate the SDE rates in finer granularity. To this end, in the next subsection we show results from the state-of-the-art microarchitecture-level fault injection framework, named GeFIN [42] which is based on the gem5 simulator.

It is essential to note that there have been several recent attempts for validating the results of microarchitecture-level models using physical accelerated beam experiments [34], [43]. Fig. 8 shows the SDE FIT rates comparison between beam and GeFIN fault injection. If the FIT rate obtained with beam experiments is higher than the fault injection the value is represented as positive; negative otherwise. It is clear from Fig. 8 that the GeFIN SDE FIT rate prediction is very close to the one measured with beam experiments.

E. *SDE Correlation to on-chip storage structures*

In this section, we examine the relationship between SDEs and the major on-chip memory structures of modern microprocessors. It is essential to evaluate the SDE rates of individual hardware structures to understand their susceptibility. Fig. 9 illustrates the susceptibility of each structure to non-benign faults that are not masked at the hardware level. An SDE can occur if a hardware error eventually becomes available at the software and silently affects the execution of the program.

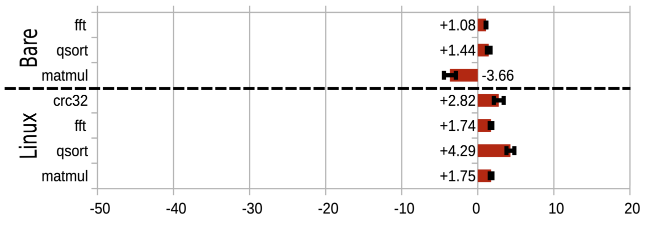


Fig. 8. Beam and fault injection SDE FIT rates comparison [34].

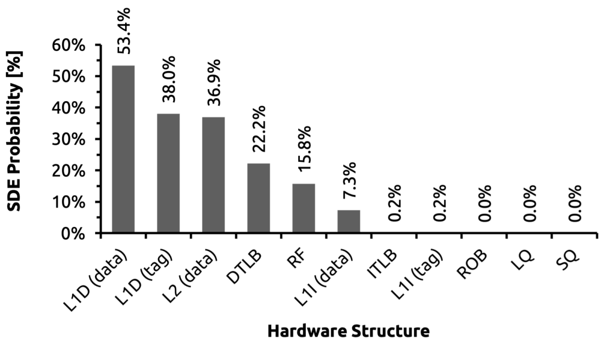


Fig. 9. The percentage of hardware corruptions at the software level (i.e., non-masked errors at hardware level) that eventually result in SDE.

Therefore, in Fig. 9 show the percentage of these errors to result in an SDE. Our first and most significant observation is that the Re-Order Buffer (ROB), Load Queue (LQ), and Store Queue (SQ) have a zero probability of experiencing SDEs. This is because any fault that occurs in these structures is not architecturally visible due to dependency graph checks that fail before the commit stage. Memory structures like the ROB, LQ, and SQ, which are deep in the microprocessor's pipeline, ensure proper instruction ordering when instructions are ready to commit. Any corruption in these structures may lead to dependency graph check failures before the commit stage and result in a crash.

V. Conclusion

The problem of silent data errors (SDEs) in today's microprocessors is a critical challenge that demands urgent attention. With the emergence of smaller feature sizes, complex computational structures, and specialized silicon features, the potential for temporary computational errors that go unnoticed during manufacturing tests is on the rise. SDEs may not be addressed by methods such as microcode updates, and can be linked to specific components within the microprocessor. Further, since the nature of these failures is silent, incorrect computation happens without any indication of an error. As such, it is crucial to develop effective methods for detecting, mitigating, and preventing SDEs in microprocessor chips to ensure reliable and accurate computing performance. We described the challenges of measuring SDE rates in both real microprocessors and simulation models (based on microarchitectural modeling on gem5). The study of SDEs is an ongoing area of research, and it is essential to continue to explore new approaches to address this critical issue in the design and implementation of microprocessors.

Acknowledgment

Adit Singh's research was supported in part by the US National Science Foundation under grant CCF-1910964. George Papadimitriou and Dimitris Gizopoulos' work has received funding in part from the EU Horizon Europe research and innovation programme under grant agreements No 101070238 (NEUROPULS), No 101093062 (Vitamin-V), and No 101097224 (REBECCA), and in part by research gifts from Meta and Intel.

References

[1] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. Debardeleben, P. C. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson, S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, and E. V. Hensbergen, "Addressing failures in exascale computing," *Int. J. High Perform. Comput. Appl.*, vol. 28, no. 2, p. 129–173, may 2014. [Online]. Available: https://doi.org/10.1177/1094342014522573

[2] C. Constantinescu, I. Parulkar, R. Harper, and S. Michalak, "Silent data corruption — myth or reality?" in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, 2008, pp. 108–109.

[3] R. Lucas, "Top ten exascale research challenges," in *DOE ASCAC Subcommittee Report*, 2014.

[4] E. Ibe, H. Taniguchi, Y. Yahagi, K.-i. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in srams from a 250 nm to a 22 nm design rule," *IEEE Transactions on Electron Devices*, vol. 57, no. 7, pp. 1527–1538, 2010.

[5] L. Bautista-Gomez, F. Zyulkyarov, O. Unsal, and S. McIntosh-Smith, "Unprotected computing: A large-scale study of dram raw error rate on a supercomputer," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 645–655.

[6] A. Chatzidimitriou, G. Papadimitriou, D. Gizopoulos, S. Ganapathy, and J. Kalamatianos, "Assessing the effects of low voltage in branch prediction units," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 127–136.

[7] P. Koutsovasilis, C. D. Antonopoulos, N. Bellas, S. Lalis, G. Papadimitriou, A. Chatzidimitriou, and D. Gizopoulos, "The impact of cpu voltage margins on power-constrained execution," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 221–234, 2022.

[8] A. Chatzidimitriou, G. Papadimitriou, D. Gizopoulos, S. Ganapathy, and J. Kalamatianos, "Analysis and characterization of ultra low power branch predictors," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 144–147.

[9] A. Chatzidimitriou, G. Papadimitriou, and D. Gizopoulos, "Healthlog monitor: A flexible system-monitoring linux service," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, 2018, pp. 183–188.

[10] D. Gizopoulos, G. Papadimitriou, A. Chatzidimitriou, V. J. Reddi, B. Salami, O. S. Unsal, A. C. Kestelman, and J. Leng, "Modern hardware margins: Cpus, gpus, fpgas recent system-level studies," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2019, pp. 129–134.

[11] A. Chatzidimitriou, G. Papadimitriou, and D. Gizopoulos, "Healthlog monitor: Errors, symptoms and reactions consolidated," *IEEE Transactions on Device and Materials Reliability*, vol. 19, no. 1, pp. 46–54, 2019.

[12] G. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, V. J. Reddi, J. Leng, B. Salami, O. S. Unsal, and A. C. Kestelman, "Exceeding conservative limits: A consolidated analysis on modern hardware margins," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 341–350, 2020.

[13] H. D. Dixit, S. Pendharkar, M. Beadon, C. Mason, T. Chakravarthy, B. Muthiah, and S. Sankar, "Silent Data Corruptions at Scale," 2021. [Online]. Available: https://arxiv.org/abs/2102.11245

[14] P. H. Hochschild, P. Turner, J. C. Mogul, R. Govindaraju, P. Ranganathan, D. E. Culler, and A. Vahdat, "Cores That Don't Count," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, ser. HotOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 9–16. [Online]. Available: https://doi.org/10.1145/3458336.3465297

[15] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[16] Y. Luo, S. Govindan, B. Sharma, M. Santaniello, J. Meza, A. Kansal, J. Liu, B. Khessib, K. Vaid, and O. Mutlu, "Characterizing application memory error vulnerability to optimize datacenter cost via heterogeneous-reliability memory," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 467–478.

[17] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, "Multi-bit upsets vulnerability analysis of modern microprocessors," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 119–130.

[18] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015, pp. 415–426.

[19] F. Hapke, W. Redemund, A. Glowatz, J. Rajski, M. Reese, M. Hustava, M. Keim, J. Schloeffel, and A. Fast, "Cell-aware test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, pp. 1396–1409, 2014.

[20] X. Lin, K.-h. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware atpg for high quality at-speed testing of small delay defects," in *2006 15th Asian Test Symposium*, 2006, pp. 139–146.

[21] C. J. Lin and S. Reddy, "On delay fault testing in logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 694–703, 1987.

[22] W. Howell, F. Hapke, E. Brazil, S. Venkataraman, R. Datta, A. Glowatz, W. Redemund, J. Schmerberg, A. Fast, and J. Rajski, "Dppm reduction methods and new defect oriented test methods applied to advanced finfet technologies," in *2018 IEEE International Test Conference (ITC)*, 2018, pp. 1–10.

[23] T. Claburn. (2021) Fyi: Today's computer chips are so advanced, they are more 'mercurial' than precise – and here's the proof. Accessed: March 2023. [Online]. Available: https://www.theregister.com/2021/06/04/google_chip_flaws/

[24] A. D. Singh, "Understanding vmin failures for improved testing of timing marginalities," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 372–381.

[25] M. D. Giles, N. Arkali Radhakrishna, D. Becher, A. Kornfeld, K. Maurice, S. Mudanai, S. Natarajan, P. Newman, P. Packan, and T. Rakshit, "High sigma measurement of random threshold voltage variation in 14nm logic finfet technology," in *2015 Symposium on VLSI Technology (VLSI Technology)*, 2015, pp. T150–T151.

[26] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE Journal of Solid-state Circuits*, vol. 25, pp. 584–594, 1990.

[27] D. P. Lerner, B. Inkley, S. H. Sahasrabudhe, E. Hansen, L. D. R. Munoz, and A. v. de Ven, "Optimization of tests for managing silicon defects in data centers," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 578–582.

[28] T. C. May and M. H. Woods, "A new physical mechanism for soft errors in dynamic memories," in *16th International Reliability Physics Symposium*, 1978, pp. 33–40.

[29] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[30] M. D. McCluskey and A. Janotti, "Defects in semiconductors," *Journal of Applied Physics*, vol. 127, no. 19, p. 190401, 2020. [Online]. Available: https://doi.org/10.1063/5.0012677

[31] G. Papadimitriou, D. Gizopoulos, A. Chatzidimitriou, T. Kolan, A. Koyfman, R. Morad, and V. Sokhin, "Unveiling difficult bugs in address translation caching arrays for effective post-silicon validation," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 2016, pp. 544–551.

[32] S. Mukherjee, *Architecture Design for Soft Errors*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[33] G. Papadimitriou and D. Gizopoulos, "Avgi: Microarchitecture-driven, fast and accurate vulnerability assessment," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 935–948. [Online]. Available: https://doi.org/10.1109/HPCA56546.2023.10071105

[34] P. R. Bodmann, G. Papadimitriou, R. L. R. Junior, D. Gizopoulos, and P. Rech, "Soft error effects on arm microprocessors: Early estimations versus chip measurements," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2358–2369, 2022.

[35] I. Tsiokanos, G. Papadimitriou, D. Gizopoulos, and G. Karakonstantis, "Boosting microprocessor efficiency: Circuit- and workload-aware assessment of timing errors," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*, 2021, pp. 125–137.

[36] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 902–915.

[37] Y. Sazeides, A. Gerber, R. Gabor, A. Bramnik, G. Papadimitriou, D. Gizopoulos, C. Nicopoulos, G. Dimitrakopoulos, and K. Patsidis, "Idld: Instantaneous detection of leakage and duplication of identifiers used for register renaming," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022, pp. 799–814.

[38] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation and Test in Europe Conference and Exhibition*, 2009, pp. 502–506.

[39] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, 2003, pp. 29–40.

[40] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[41] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, aug 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[42] A. Chatzidimitriou and D. Gizopoulos, "Anatomy of microarchitecture-level reliability assessment: Throughput and accuracy," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2016, pp. 69–78.

[43] A. Chatzidimitriou, P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "Demystifying soft error assessment strategies on arm cpus: Microarchitectural fault injection vs. neutron beam experiments," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 26–38.