Chip-Level Soft Error Estimation Method

Hang T. Nguyen, *Member, IEEE*, Yoad Yagil, Norbert Seifert, *Senior Member, IEEE*, and Mike Reitsma

Invited Paper

***Abstract*—This paper gives a review of considerations necessary for the prediction of soft error rates (SERs) for microprocessor designs. It summarizes the physics and silicon process dependencies of soft error mechanisms and describes the determination of SERs for basic circuit types. It reviews the impact of logical and architectural filtering on SER calculations and focuses on the structural filtering of soft radiation events by nodal timing mechanisms.**

***Index Terms*—Detected uncorrectable error (DUE), failure in time (FIT), logic derating (LD), mean time to failure (MTTF), silent data corruption (SDC), single event upset (SEU), soft error rate (SER), timing derating (TD).**

I. Introduction

SOFT ERRORS have been extensively studied in the decades since the discovery of their impact on memory cells and logic circuits [1], [2]. Early studies dealt with the physics of the phenomenon, examining and modeling the interaction of particles with very large scale integration (VLSI) structures. Later studies sought to determine the error rate of specific circuits and to define process trends. Much work was also dedicated to soft error prevention, focusing on process dependencies, circuit hardening, fault tolerance, redundancy, and error correction schemes. The IBM Research Journal Special Issue on Soft Errors (January, 1996) provides a compelling summary of the work done at IBM from the beginning of that company's work on soft errors in VLSI circuits [3]–[8].

Initially, soft error rate (SER) studies were focused on memory cells, both dynamic and static. While such circuits are susceptible to soft errors, they can be effectively protected by error correction schemes. Lack of an effective error correction technique leads to an increasing focus on the SER of dynamic and static logic. Baumann [9] discussed technology scaling and concluded that failure in time (FIT)/bit saturates for memory devices but system SER continues to increase as the number of susceptible structures per design grows. He predicted that sequential logic SER would become the reliability limiter once SRAMs were protected. Cohen *et al.* [10] examined the importance of the SER of static logic, predicting that it would become comparable to that of dynamic circuits. Shivakumar *et al.* [11] also examined the SER impact of logic circuits and predicted that logic SER per chip would increase by nine orders of magnitude from 1992 to 2011, becoming comparable to the failure rate of unprotected on-chip memory.

Estimating the probability of soft errors in actual circuits requires the modeling of the charge injection as well as the circuit response. Srinivasan *et al.* [12] developed a soft error modeling and measurement (SEMM) tool that addresses the entire process from modeling the interaction of particles with silicon to circuit response. In this approach, SPICE simulations are performed to analyze the circuit response to a soft error event and a particle hit is simulated by a current source that models the particle type and energy [13], [14].

The results of this kind of evaluation reflect only part of the analysis that must be completed to model the propagation of a soft radiation event to a sequential holding the final logical state of a device. A comprehensive estimate of the SER of a product must take into account not only fundamental physical phenomena but also the structural filtering of soft radiation events due to logical roadblocks, architectural features, and nodal timing. This paper draws upon both new and previously published work to summarize the considerations required to complete such an estimate [15]–[17].

This paper is organized as follows. Section II describes the origin of alpha and neutron particles and their interaction with silicon to generate soft errors. Section III describes the concept of the propagation of soft errors to sequentials holding machine states and nonmachine states. Section IV describes the soft error types and how soft errors are specified in products. The concepts of nominal FIT, timing derating (TD), and logic derating (LD) are examined in Sections V–VII, respectively. Section VIII describes how to roll up a chip failure rate both at the beginning and at the end of the design cycle.

II. Physics of Soft Errors

Soft errors are intermittent hardware malfunctions that are not reproducible. If one resets the system and reruns the software, the hardware will respond properly. Soft errors introduced by radiation events are generally due to either alpha particles [1] or neutrons [2]. An updated and in-depth review on soft error physics and the interaction of alpha and neutron particles with silicon is provided in Baumann's article in this special issue [18].

Alpha particles originate from residual radioactive atoms in a chip's packaging material. They interact with the silicon lattice through both kinetic and Coulomb forces, generating electron–hole pairs along their paths. When an alpha particle hits a junction area, drift and diffusion mechanisms result in the

Manuscript received February 14, 2005; revised July 7, 2005.

The authors are with Intel Corporation, Hillsboro, OR 97124-6009 USA (e-mail: hang.t.nguyen@intel.com).

Digital Object Identifier 10.1109/TDMR.2005.858334

collection of these carriers by the junction [18]. If this carrier current is large enough, the impacted node can transition to a new logical value. Storage nodes in latches and memory cells will retain this new value until they are rewritten. Nodes in static logic will return to their previous state in a few picoseconds, but the nodal transition may generate a logical disturbance that propagates on to impact sequentials holding the machine state. The probability of generating a given level of carrier charge is dependent on the spectrum of the alpha flux as well as the junction's area and shape. Detailed studies of carrier current waveforms can be found in [12] and [13]. Technology advances over the last few years have enabled the manufacture of materials with extremely small radioactive residues, significantly reducing the contribution of alpha particles to product SER.

The cosmic ray flux interacts in the atmosphere to generate cascades of protons, pions, muons, and neutrons. Neutrons have no charge; hence, their interaction with the silicon lattice is purely kinetic. They dominate the interaction with silicon under terrestrial conditions; but in contrast with alpha particles, most neutrons do not interact with Silicon—only a small portion of the neutron flux generates electron–hole pairs. Neutrons tend to generate more charge than alpha particles. Typical waveforms reflect tens of fempto-Coulombs generated within a period of a few picoseconds [12]. Details on the interaction of neutrons with silicon are provided in [8] and [18].

III. Error Cases in a Typical Microprocessor

A. Machine States and Nonmachine States

The logical state of a CPU consists of two primary entities: machine states and nonmachine states [15], [19], [20]. Machine states are those components of the logical structure that are designated as critical and represent the "state" or "signature" of the machine. These machine states translate to user-visible events at the terminals of the processor. Latches holding these states must be restored to a logically correct state to restart the machine after an interruption. Machine states include architectural and memory states. Architectural states include data held in register files (integer and floating point) and application and control registers that specify the operating mode of the machine. The device's outputs represent machine states since they are directly visible to the system. Memory states include data held in caches (typically at several levels) and translation look-aside buffers (TLBs). Memory states are included among machine states since any error in these states will also result in a change in the machine's behavior. Machine state is held in two circuit forms: structured arrays (caches, register files) and random state elements (application and control registers).

Nonmachine states are held in supporting circuits surrounding sequentials holding machine states. Sequentials holding nonmachine states by themselves are not critical but act as a conduit to propagate an error to sequentials holding machine states. Sequentials holding nonmachine states include latches (control and data path) and combinational logic (also control logic and data path logic). Combinational logic is typically made up of static CMOS gates and dynamic CMOS (domino) circuits. The clock distribution network generally contains non-

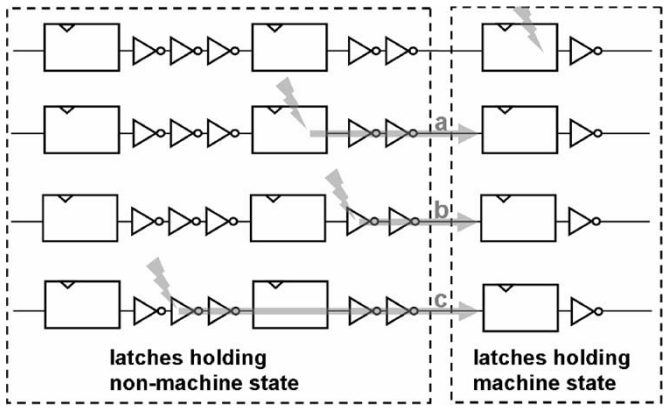

Fig. 1. Error propagation through logic and latches holding machine and nonmachine states.

machine state latches. The storage of nonmachine states also occurs in two classes of sequentials: structured arrays (such as branch prediction structures) and random storage elements.

B. Error Cases in Machine and Nonmachine States

Soft errors occurring in a microprocessor can be classified into two main categories: errors in sequentials holding the machine state and errors in sequentials holding the nonmachine state. Fig. 1 shows the possible error cases. The first case involves a strike occurring directly in sequentials holding the machine state, shown as a strike in the latch in the upper right corner of Fig. 1. However, not all errors in sequentials holding the machine states result in machine failure. For example, an error in an invalid line of a cache is benign since an invalid line is not used by the processor.

The second case includes strikes in sequentials holding the nonmachine states. Some errors will evaporate while others can propagate into the sequentials holding machine states, causing machine failure. Three conditions exist that could propagate errors from sequentials holding nonmachine states into those holding machine states. These are shown as a, b, and c in Fig. 1. Condition a) depicts an error in the latches; this error can then propagate to the sequentials holding machine states. Condition b) shows an error in the combinational logic (static or domino); this error similarly can propagate to the sequentials holding machine states. Condition c) also shows an error in the combinational logic. However, this error first propagates to the latches holding nonmachine states and then to the latches holding machine states.

Not shown in Fig. 1 are errors in the latches and combinational logic that simply vanish before reaching any latch holding machine state. In other words, not all errors that originate in latches holding nonmachine states will propagate to latches holding machine states to cause machine failure. For example, a branch flush due to a branch mispredict would inhibit any error in the pipeline being flushed from ever reaching and corrupting the machine states. It is rare for an error reaching a machine state to not cause a machine failure. This would be possible only if the error is overwritten with good data prior to being used or the error is neutralized by another soft radiation event. For practical purposes, any error originating from sequentials

holding the nonmachine state and reaching sequentials holding the machine state will cause a machine failure.

These considerations lead to the notion of a soft error derating factor, defined as the ratio of actual machine failures to the number of soft radiation events (radiation-induced nodal upsets) impacting a product. A machine failure is defined as a change in the normal behavior of the machine, including undetected data corruption, machine hangs, and nonrecoverable errors. A product failure rate is the sum of all element failure rates for the design, including contributions from sequentials holding machine and nonmachine states. Derating includes two components, TD and LD. It is critical to understand and estimate TD and LD appropriately in order to complete an accurate assessment of the predicted FIT rate of a microprocessor design.

IV. Soft Error Classifications

A. Soft Error Types

An SER-induced error can be categorized into one of four types, depending on how the system responds to the error: 1) masked errors; 2) correctable errors; 3) detected uncorrectable errors (DUE); and 4) silent data corruption (SDC). Masked errors are those errors that get "erased" by the machine; they disappear and do not cause any harm. An example would be an error in a pipeline latch, but the pipeline gets flushed due to a branch mispredict, thus making the error irrelevant. Correctable errors are those detectable and correctable by either hardware or software. An example is an error detected and corrected by the built-in error correcting code (ECC) hardware. DUEs are those errors that are detected but not correctable such as those in memories protected by parity but without error correction. These errors can result in a system hang or an application termination, depending on the severity of the error. DUE affects the system availability or system uptime. Lastly, SDCs are those errors that are not detectable and can silently corrupt the system data. An example includes an error in the data path that eventually gets written back to memory without the user knowing. These are the most catastrophic types of error and will affect the system's data integrity or reliability.

B. Soft Error Specifications

SERs are often specified at the system level and can vary depending on system applications. For instance, SER goals for a server used in a banking system will be much more restrictive than those for a cell phone. SERs are often specified in mean time to failure (MTTF) rates. The IBM Server Group specifies an MTTF rate of 1000 years for SDC, 25 years for DUE errors causing system termination, and 10 years for DUE errors causing application termination [21]. One common measure of soft errors rate is FIT, where one FIT is one failure per billion device hours, or approximately one failure per 100 000 years (114 155 years to be precise). FIT rate is dependent on the operating condition under which the device operates including voltage, temperature, altitude, etc. The 1000-year SDC and 10-years DUE MTTF goals roughly map to 100 FIT for SDC and 10 000 FIT for DUE. Meeting these error rates will be a challenge to any design. The above specifications are defined per system, not per component; therefore, systems with multiple CPUs and additional components need to budget the FIT rate between all components. The SDC goal is typically ten times more stringent compared to DUE goal due to its catastrophic nature. The sum of SDC and DUE FIT is usually referred to as the SER of a chip.

Three components make up the estimated FIT rate of any given circuit element.

1) Nominal FIT: the probability of a single event upset (SEU) occurring on a specific node. This depends on circuit type, transistor sizes, node capacitance, VDD value, temperature, and the downstream path in case of nonrecycled circuits. It also depends on the state of the inputs of the driving stage and the probability of each input vector, often referred to as the signal probability (SP) of the circuit.
2) TD: the fraction of time in which the circuit is susceptible to SEU that will be able to propagate and eventually impact a latch holding the machine state (to be elaborated later).
3) LD: the probability of an SEU to impact the behavior of the machine. It is dependent on the applications as well as the microarchitecture of the device. We find that a significant fraction of soft errors in a processor actually vanish without causing any misbehavior of the machine.

The FIT rate of each element is the product of these terms:

$$\text{Element FIT} = \text{Nominal FIT} \times \text{Timing Derating} \times \text{LD}. \quad (1)$$

This simplification enables a systematic and effective method to be developed for deriving the chip FIT rate by breaking the chip down into three components that can be independently derived using tools developed to estimate their values. In reality, there are interactions between nominal FIT and TD that make it difficult to separate these two components as independent variables. The resulting interactions can be consolidated into the nominal FIT component or the TD component once the first-order FIT/TD/LD analysis is complete.

V. Nominal FIT Estimation

As noted above, we define nominal FIT rate as the probability of an SEU occurring on a specific node. This depends on circuit type, transistor sizes, node capacitance, VDD value, temperature, and the downstream path in case of nonrecycled circuits. It also depends on the state of the inputs of the driving stage and the probability of each input vector, often referred to as the SP of the circuit. Nominal FIT does not include logic or timing derating (discussed in detail below). This definition is practical since it enables a simple summation of the raw FIT rates of all subcircuits in a design. Attempting to separate the nominal FIT from signal probabilities would lead to a large number of FIT rates per subcircuit. We state that separating the circuit response to nominal FIT and TD components is a simplification. Accurate modeling using circuit simulation of the entire circuit along a particular path results in an FIT

rate that most accurately represents both nominal and TD contributions.

Nominal FIT estimation is mainly based on SPICE simulation [13], [14]. The critical charge model is a useful and reasonably accurate approach in which a uniform waveform is used for each particle type, and only the magnitude of the peak of the injected current is modified. Several iterations are used to find the critical charge $Q_{\rm crit}$, for which the circuit fails (say a latch is flipped and stores the wrong value). Particles that generate less charge than $Q_{\rm crit}$ are considered harmless, and particles that generate more charge are considered to cause a soft error. Once $Q_{\rm crit}$ is defined, one needs to perform probabilistic analysis to convert $Q_{\rm crit}$ to nominal FIT—the probability of the circuit under question to be hit by a particle and fail. Since neutron and alpha particles have different waveforms, they have different $Q_{\rm crit}$ values. Moreover, since the transistors respond to the injected charge differently, the $Q_{\rm crit}$ value for $1 \rightarrow 0$ is different from that of $0 \rightarrow 1$ transition. So, each circuit gets four $Q_{\rm crit}$ numbers, indicating its electrical susceptibility to $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions due to alpha and neutron strikes. This also implies that the nominal FIT per specific circuit is not well defined: it depends on the fraction of time the circuit is susceptible to $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions caused by alpha and neutron hits.

Virtually all circuit types, including memories, latches, flip-flops (FF), static and dynamic combinational logic, and data path and control logic, are subject to SEU. The failure of a recycled node is well defined since it can "flip" and store a wrong value. Other circuits do not fail directly, but could cause a downstream recycle node to store a wrong value. Therefore, all circuit types, including domino and static CMOS, and all parts of the chip are considered possible sources for soft errors. The following sections describe how to estimate the nominal FIT of three circuit types: latches, domino gates, and static logic gates. These three types cover the main concepts of how to estimate FIT rates and can be generalized to most other circuit types.

Another failure mode, which is not discussed here, is the SER-driven time-based jitter on nodes with large capacitance, where the particle hit does not cause a full swing glitch. For example, a particle hit on a global clock net could result in some jitter that is sufficient to cause failure [22]. A similar sensitivity is expected on SRAM bit lines, where a relatively small noise (far less than full swing) could result in malfunction.

A. FIT Estimation of a Latch

The term latch here represents any storage element built of a recycle loop: a storage node and a feedback node. Moreover, its nominal FIT is only defined for the period at which it is in stored mode (as opposed to transparent mode). In this context, we only analyze the recycle loop of the latch. Nominal FIT estimation includes the following steps.

1) Find $Q_{\rm crit}$ values using circuit simulation: inject waveforms of alpha and neutron particle hits on all relevant nodes and scale the deposited charge $Q_{\rm inject}$ until a failure is detected (that is, the storage node flips). The minimal injected charge that causes failure is defined as $Q_{\rm crit}$, the critical charge that causes failure. Waveforms with injected charge smaller than $Q_{\rm crit}$ will cause a temporal glitch on the hit node but the circuit will recover and keep its original logic value. Waveforms with injected charge larger than $Q_{\rm crit}$ would cause a wrong value to be latched. Note that a latch has multiple $Q_{\rm crit}$ numbers per node, as determined by the polarity of the node voltage and the impacting particle type.
2) For each diffusion, calculate the FIT rate based on the $Q_{\rm crit}$ of the node connected to it and diffusion details (area and geometry). This stage takes into account the spectrum of alpha and neutron flux at the conditions specified by the user, as well as averaging over all possible strike locations and angles within the specified diffusion.
3) Compute the weighted average of the $1 \rightarrow 0$ and $0 \rightarrow 1$ FIT rates based on the SP of the storage node.

A simple jam latch includes two nodes, the storage and the recycle nodes; hence, one needs to simulate charge injection only to those two nodes. Other latches include additional circuitry that also needs to be considered—an example is a latch with tristate feedback gate to avoid contention currents during write operation. For such circuits, one also needs to simulate charge injection to internal nodes and find their own $Q_{\rm crit}$ values. Since each $Q_{\rm crit}$ value characterizes the circuit sensitivity to SEU at certain polarity and certain logic state, separating nominal FIT from signal probabilities is not practical.

B. FIT Estimation of a Domino Gate

A domino gate is a dynamic circuit operating in two phases: precharge and evaluation. Data in the evaluation phase are maintained by a half or full keeper. Therefore, SER analysis of a domino gate is similar to that of a latch: the circuit can fail due to SEU and will not recover until the next precharge phase. Indeed, nominal FIT estimation follows the same steps as latch analysis. There are two main differences: 1) a domino gate should only be analyzed for one type of failure: $1 \rightarrow 0$ in n-logic domino (which is the widely used domino type) and $0 \rightarrow 1$ in p-logic domino and 2) domino has several inputs; hence, the total amount of circuit configurations is larger than in a latch. This leads to multiple simulations up to 2^n, where n is the number of inputs.

Consider, for example, an n-type domino gate as shown in Fig. 2. A strike on $\rm n_1$ can cause the gate to fail if either a, b, or both are at logic 1; similarly, a strike on $\rm n_2$ will have effect if c is also in logic state 1. The state of inputs a–c also affects the effective capacitance seen on node "out," and the impedance between $\rm n_1$ and out is lower if both a and b are at logic 1 as opposed to just one of them. The nominal FIT rate of a domino cell is thus dependent on the probability of each input to be at logic state 1. More precisely, it is dependent on the probability of each input vector that in turn could be estimated by the SP of its inputs and output.

C. FIT Estimation of a Static Gate

Static gates are similar to domino gates except that there are no recycle elements; hence, the outputs will always recover. That is, a static gate, by itself, cannot fail in the sense of

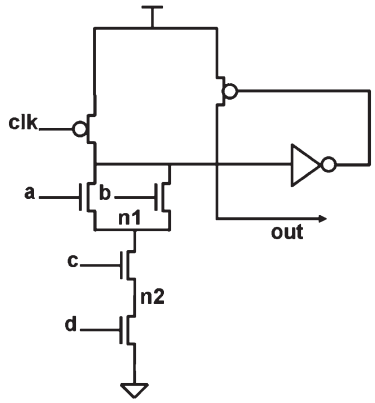

Fig. 2. Sample n-type domino gate (from [15]).

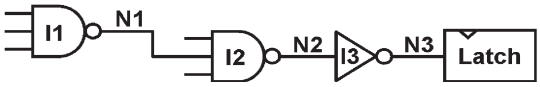

Fig. 3. Sample static logic path (from [15]).

permanent logic state flip. Instead, a static gate can suffer a glitch that might propagate downstream and permanently flip a latch or a domino gate. Permanent here means until the next event, namely the next clock. We have studied in detail an 8-bit CLA adder and compared the SER contribution due to static node hits in the adder logic to the nominal and derated SER of the receiving sequentials. Our results show that for the studied technology, the combinational contribution equals up to 30% of the nominal SER of the receiving sequentials and up to 100% of the derated SER of the sequentials. The exact value depends on the input vectors as well on VDD and the clock frequency for a given technology. The dependence on the clock frequency is linear, as will be shown below. This clearly underscores that the static combinational SER cannot be neglected for a high-speed microprocessor.

Fig. 3 shows a logic chain ending in a latch. A particle hit on gate I_1 will generate a glitch on node N_1, which could propagate through gates I_2 and I_{33} and then be latched. This is the mechanism of an SEU originating in a static gate. The nominal FIT of gate I_1 is the probability of an SEU due to a particle hit on I_1 when surrounding conditions enable the glitch propagation, namely, the other two inputs to gate I_2 in the above example are at logic "1" and the particle hits I_1 at such timing that the SEU glitch arrives at the latch during its setup and hold time.

Estimating the nominal FIT of all static gates in a chip by simulating their full paths would be a compute-intensive task. One must adjust logic and timing conditions to enable propagation and latching SEU-induced glitches. A more practical approach is to estimate the glitch size that would be able to cause a wrong value in the latch and simulate a much smaller circuit: just the static gate. Heuristics can be used to estimate the critical glitch width W_{crit} measured at a certain voltage threshold V_{trip}. A glitch with $W < W_{\mathrm{crit}}$ will not affect the latch. A glitch with $W > W_{\mathrm{crit}}$ is able to fail the latch. Once W_{crit} and V_{trip} are defined, circuit simulations are used to translate these numbers into Q_{crit} and FIT rates, in the same

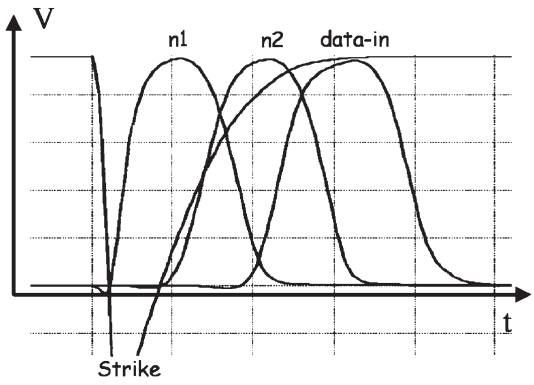

Fig. 4. Glitch propagation (from [15]).

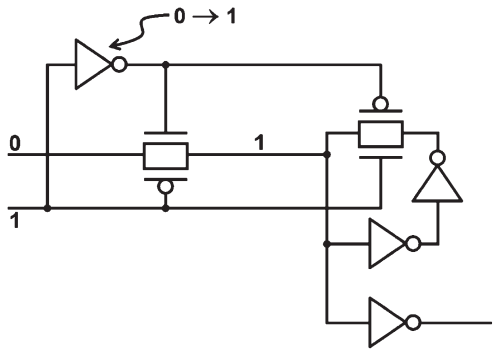

Fig. 5. Sample latch (from [15]).

way as done for domino circuits. We therefore define a failure criterion for static gates as a glitch larger than W_{crit} measured at a certain threshold voltage V_{trip}. The difference between this failure criterion and that of latch and domino is that the actual values depend on the downstream path rather than on the static gate itself.

Fig. 4 shows the simulated glitch propagation through an inverter and two buffers into the data-in pin of a latch. The negative peak is the particle strike that causes voltage undershoot (deposited charge is larger than the charge stored in the node). The next three glitches indicate that the SEU on the hit node propagates through the logic path and generates full swing glitches that are neither evaporated nor attenuated along the path. The glitch on the data-in pin of the latch is captured by the latch and causes the latch to fail (read wrong data) when the clock edge is adjusted to the glitch timing. Sensitivity to clock edge timing (relative to the glitch) is moderate: on the order of the glitch width. That is, as long as there is partial overlap between the glitch and the clock edge, wrong data are latched.

Static gates that drive domino ones are similarly analyzed. In this case, V_{trip} would be the n-ch threshold voltage and W_{crit} is the pulse width needed to flip a domino. Since domino circuits are designed for fast response, static gates that drive domino are likely to have high nominal FIT rates.

Static FIT is not limited to data paths. Control logic is also sensitive to SEU, and the analysis is similar to the one described above. For example, consider a latch in "stored" mode as shown in Fig. 5.

SEU on the control/clock inverter opens the n-ch transfer gate and simultaneously closes the p-ch transistor in the recycle

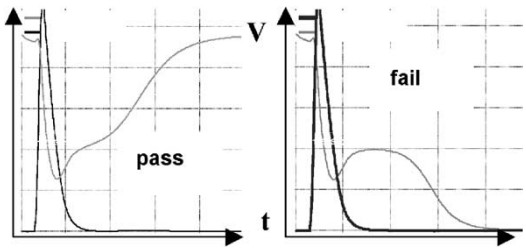

Fig. 6. Latch waveform as a response to direct hit on a control signal causing a false write. The sharp peaks (in black) indicate the control node response to charge injection; the lighter waveforms show the voltage response of the data node within the latch (from [15]).

circuit, which holds the logic 1 value on the storage node. As a result, a false write operation takes place while the latch is supposed to be in "stored" mode. Similarly, a particle hit during write operation will cause a jitter on the control logic, shift the exact timing of the write operation, and hence might lead to latching of wrong data.

Depending on the latch and FF structures, the nominal FIT contributed by control logic could be comparable to that of the recycled (data) nodes. When deriving the nominal FIT rates of library cells, one should consider both data path and control logic in the FIT rates of storage cells. Fig. 6 presents sample waveforms for the recovered and failed cases due to a hit on its control logic that causes a false write. The difference between Q_{inject} of the two strikes shown below is 5%.

D. Scaling Trend for Nominal FIT

Baumann [9] discussed technology scaling and concluded that FIT/bit saturates while system SER increases since the number of cells per chip is rapidly growing. Bossen [21], Karnik *et al.* [23] resent process scaling data, showing that neutron SER per latch is likely to be constant in future processes if the supply voltage continues to scale by only 0.8 times. Hareland *et al.* [24] studied process scaling on SRAM and the differences between bulk silicon and silicon on insulator (SOI). They conclude that FIT rate per bit reduces for future technologies. Partially depleted SOI is found to have lower SER sensitivity compared with bulk silicon in 0.25-μm technology; however, in 0.18 μm, the two technologies indicate similar SER sensitivity. Hazucha *et al.* [25] conducted neutron SER measurements on advanced 90-nm logic technology. They found that the FIT rate increases by 18% per 10% voltage reduction and that FIT is linear with diode area. This results in moderate reduction of FIT/bit per generation. This study also finds that n-diffusion is ~ 2 times more sensitive to neutron strike than p-diffusion. Dodd *et al.* [26] performed SER measurements on SRAMs produced by several vendors and found large differences in FIT rate between different vendors at the same technology generation. This emphasizes the sensitivity of SER to details of the process and the design. Dodd *et al.* also reported neutron-induced latch-up, which in some cases was so dominant that they prevented SER rate measurements. This is an important finding as error correction measures cannot overcome latch-up failures observed in some technologies. A comprehensive process scaling discussion is given by Baumann [18] in this special issue on soft errors.

The main factors controlling the scaling of nominal FIT are as follows.

1) Diffusion area—the probability of SEU on a specific node is roughly proportional to the area of diffusions of that node since charge separation occurs at or near the diffusions [25]. Typically, this area is reduced by two times per generation; hence, FIT per cell should decrease. Note that typically the number of transistors (or cells) is doubled per generation so that the FIT rate per product tends to be constant from the diffusion area point of view.
2) Charge scaling—a simple optical shrink results in lower capacitance per node. In addition, VDD is reduced per generation, resulting in even less charge per node. This tends to increase the sensitivity to soft errors as low energy particles that generate less charge are able to flip more nodes. Charge scaling dominated the SER trend at older processes: SER sensitivity increased per generation. However, in deep submicron technologies, many circuits such as memory cells are flux limited or saturated. That is, any particle hit will cause SEU. In such a case, process scaling reduces the diffusion area (see 1 above) but does not increase the sensitivity to SEU; hence, overall FIT per cell is reduced for cells that are flux limited.
3) Voltage scaling—as mentioned in 2) above, voltage scaling has historically tracked process dimension scaling and has contributed to the trend to increased soft error sensitivity (SES) with process evolution. But in recent process generations, voltage scaling has lagged process dimension scaling, contributing to a decline in FIT per bit for the 90-nm and 65-nm technologies.
4) Process advances—such as SOI or similar partial or fully depleted layers significantly reduce the charge collection volume and/or efficiency leading to lower sensitivity to soft errors. SER sensitivity is also impacted by details of doping profiles and doses, where a clear trend cannot be defined. One has to measure and/or simulate charge collection efficiency and typical charge injection waveforms per process. IBM reported a five times improvement in FIT rate for partially depleted SOI for SRAM cells at 90 nm with no data reported on latches [24], [27].
5) Flux of alpha particles—strongly depends on the amount of radioactive residues and details of the back end (metal layers). Cleaner materials and more metal layers tend to reduce the flux; hence, alpha particles become a lesser issue in modern processes. Note, however, that alpha particles impact nodes with very small charge; hence, sensitivity to alpha particles increases per generation except for flux-limited circuits.

Technology trends of SRAM cells and a specific latch are shown in Fig. 7. The FIT rate per bit of these cells increased from 180-nm to 130-nm technologies and decreased in 90-nm and 65-nm processes. Note that SER sensitivity per product still increases per generation as the number of cells is roughly doubled per process while FIT per bit only gradually reduces when no additional protection is added.

Summing up the process scaling factors, one finds a nontrivial and nonmonotonic behavior. Memory cells and many other

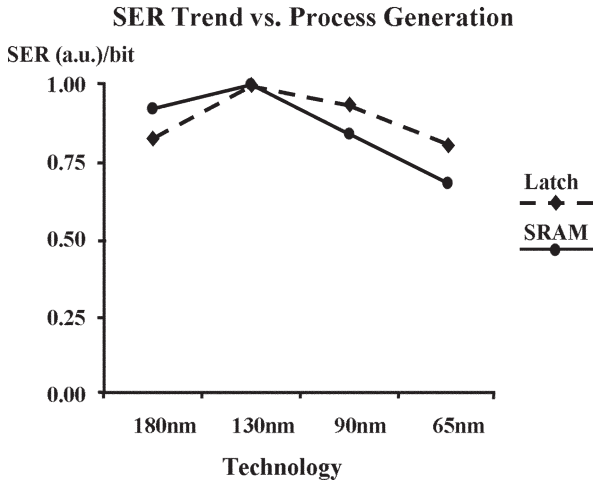

Fig. 7. Technology trend of an SRAM and a latch cell, each normalized to the corresponding SER value at 130 nm.

circuits that mainly include minimum or close to minimum size transistors are flux limited and their FIT per cell reduces per generation (in deep submicron technologies). Circuits with high capacitance that were immune in previous processes are becoming susceptible and will see an increase in their FIT rates. This leads to the need to address all circuits in a product SER analysis rather than analyzing only the "sensitive" circuits.

VI. TD Estimation

TD denotes the fraction of time a node is susceptible to upsets. TD therefore accounts for the circuit environment and the dynamic biasing that are not considered in estimating nominal FIT rate.

A. TD of Sequential Circuits

The TDs of sequentials in memory arrays whose contents are typically stored for multiple clock cycles are usually conservatively set to 1.0 (no derating) and not further discussed here. In the following, we will focus on the TD of sequentials implemented in a pipeline. In that case, any fault has to propagate to the receiving sequential or memory element further downstream to cause an error. The TD of sequentials in a pipeline therefore depends on the delay to the next downstream receiving memory element and on the cycle time T_{cycle} of the pipeline. For FFs, the TD strategy has been characterized and described in detail by Seifert and Tam [16].

For sequentials holding nonmachine states, TD depends on the type of sequential as well as on the logic path length of each stage in the data or control path. The time during which a latch is vulnerable to soft radiation events (its T_{WOV} or "window of vulnerability") is limited to its "hold" (nontransparent) time. Fig. 8 shows how the window of vulnerability of the slave latch in a master–slave FF (MSFF) is reduced by the delay from its output to the next MSFF in the data path. This is because the last instant in the slave latch's window of vulnerability for which an SEU can still propagate to the input of the next MSFF before it closes is reduced linearly by the delay to the next MSFF (plus the MSFF's setup time and clock skew and jitter). If the fault

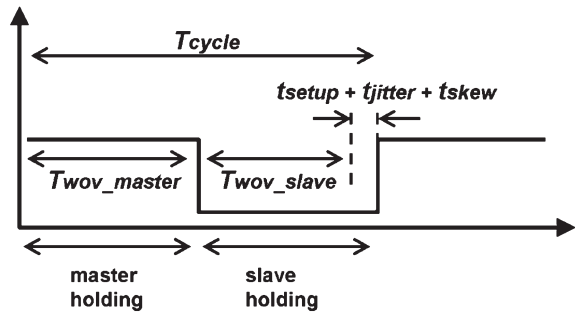

Fig. 8. Timing governing the TD of a MSFF. Any upset occurring outside the sensitive time window T_{WOV} will not propagate in time to the next downstream sequential and therefore will not contribute to the SER. The propagation delay impacts the T_{WOV} of the master only for $T_{\mathrm{prop}} \geq T_{\mathrm{phase}}$. Figure from [16].

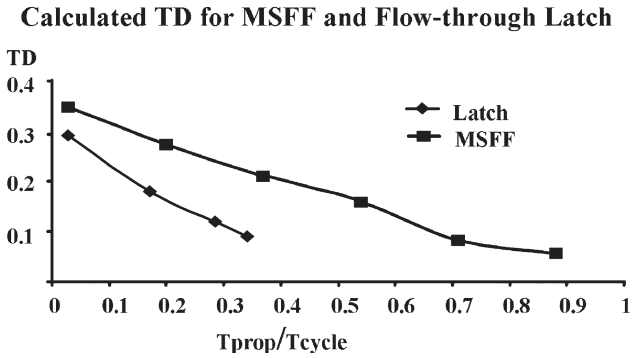

Fig. 9. Calculated TD for a master–slave FF and a "flow-through" latch as a function of the relative propagation delay. TD decreases with increasing propagation delay. The offset at $T_{\mathrm{prop}} = 0$ is due to the intrinsic delay in the sequentials. Figure from [16].

arrives too late, it will not be latched and will not cause an error. A general description of the determination of the TD of flow-through latches is given in the next section. Fig. 9 depicts the TD of an MSFF and a "flow-through" (transparent) latch as a function of the relative propagation delay of the latch/FF to the downstream sequential (the TD of the MSFF is plotted as the sum of the master and slave TD factors). Fig. 9 shows that at slow clock frequencies the value of TD approaches 0.5 for the MSFF, denoting the case of SEUs occurring during the master and slave windows of vulnerability, each having adequate time to propagate to the next MSFF.

The TD of MSFFs can be modeled qualitatively using

$$\mathrm{TD}^{\mathrm{FF}} = 0.5\frac{\left(T_{\mathrm{cycle}} - \left(\Delta t_{\mathrm{tot}}^{\mathrm{prop}} + \Delta t^{\mathrm{clk}}\right)\right)}{T_{\mathrm{cycle}}} = 0.5\frac{\left(T_{\mathrm{WOV}}\right)}{T_{\mathrm{cycle}}} \tag{2}$$

where

- $\Delta t_{\mathrm{tot}}^{\mathrm{prop}}$ sum of the propagation delay through the combinational logic and the intrinsic delay within the sequential;
- Δt^{clk} impact of setup time, clock rise and fall times, and clock skew;
- T_{WOV} "window of vulnerability" and equals the effective time window when the upset will be latched downstream.

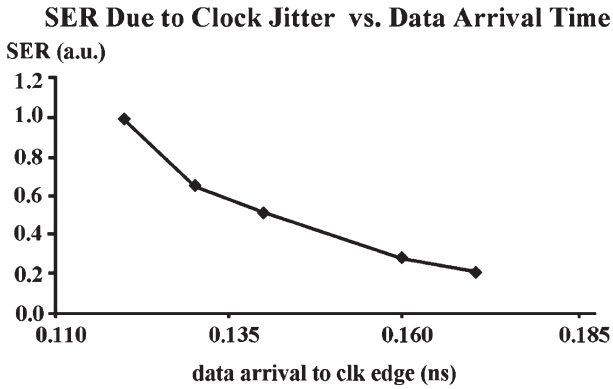

Fig. 10. Failure rate due to upsetting clock buffer nodes is plotted as a function of time between data arrival time and the time when the clock signal is asserted. One can clearly observe the approximately exponential nature of radiation-induced clock jitter.

The 0.5 factor is specific to MSFFs and reflects the fact that at slow clock speeds the master and slave are at maximum 50% of the time susceptible (i.e., not driven). Note that upsets of clock nodes local to the sequentials are not accounted for in (2). The main reason for this is that the SER and TD of radiation-induced jitter depend on the arrival time of the data.

Fig. 10 shows the strong dependence of the SER of clock node hits on the data arrival time with respect to the clock edge at the receiving FF. This increase is mainly due to the fact that when the data arrive just a setup time before the clock asserts, only very little charge is necessary to move the clock edge to yield a setup time violation. Another clock node-induced upset mode is a false opening of the sequential, which is equivalent to the scenario discussed in connection with Fig. 5. Particle-induced false writes in control logic can also be called radiation-induced race, since the data get falsely written into the next sequential and potentially race through the next pipeline stage. Clock jitter and race are not discussed any further in this paper in the context of TD factors and will be analyzed in a subsequent publication [22].

B. TD of Flow-Through Latches in Latch-Based Data Paths

The analysis of the time derating for a "flow-through" latch in a latch-based data path design is different than that for latches in an FF or pulse-latch-based design. In an FF or pulse latch data path design, the TD analysis examines only the single cycle of the path following the sequential impacted by the radiation event. In a flow-though latch data path design, a data transition due to the propagation of a radiation event to the input of a latch during its transparency causes the TD analysis to be continued to the next latch in the path. If the data transitions due to the radiation event propagate to this latch during its transparency, the TD analysis continues to the next latch, and so on. This process is analogous to the calculation of timing margin in a "flow-through" latch-based data path design, in which timing margins are determined by breaking up data paths into multilatch chains that only terminate when the worst case data transition at the input of the next latch in the chain occurs before the beginning of transparency for that latch. The lengths of these paths vary depending on the exact sequence of delays between latches in the chain and may be many phases long. Similarly, TD analysis of a flow-through latch in this kind of data path can require the examination of a sequence of many interlatch delays. The formulas and discussion below describe an iterative procedure that determines the TD for a flow-through latch as a function of each interlatch delay in the path from the radiation-impacted latch to the point in the latch sequence where the analysis terminates.

This procedure defines the "window of vulnerability" (T_{WOV}) of a latch as the time during which a radiation event impacting the given latch has the potential to propagate to a latch holding a machine state barring logical or architectural roadblocks [15]. For an MSFF, the window of vulnerability of the master latch is the entire first phase of any given cycle if the delay to the next MSFF in the data path is zero and diminishes linearly with the interflop delay for delays greater than one phase [16]. For a flow-through latch, the window of vulnerability is potentially as long as its entire interval of nontransparency (one phase) and diminishes in a manner dependent on all the interlatch delays in the path. The window of vulnerability defined by analysis of the first n latches in the path following the impacted latch is defined as $T_{\mathrm{WOV}(n)}$.

The number of latches analyzed before terminating the algorithm is dependent on the "critical phase" of each latch in the path. This is the phase (counted from the phase containing the window of vulnerability of the radiation-impacted latch) during which the inputs of a latch following the impacted latch must be driven if an SEU is to continue to propagate. If the SEU-induced path propagating from the last instant of $T_{\mathrm{WOV}(n-1)}$ arrives before the nth latch's critical phase, the algorithm ends and $T_{\mathrm{WOV(final)}} = T_{\mathrm{WOV}(n)} = T_{\mathrm{WOV}(n-1)}$ and the final value of TD for the impacted latch is $T_{\mathrm{WOV(final)}}/T_{\mathrm{cycle}}$. If the radiation-induced path propagating from $T_{\mathrm{WOV}(n-1)}$ arrives during the nth latch's critical phase, the value of T_{WOV} also remains the same ($T_{\mathrm{WOV}(n)} = T_{\mathrm{WOV}(n-1)}$), but the algorithm continues to the next latch. If the delay from the last instant of $T_{\mathrm{WOV}(n-1)}$ to the input of the nth latch puts the transition after the critical transparency of the nth latch, a new smaller value for $T_{\mathrm{WOV}(n)}$ is determined ($T_{\mathrm{WOV}(n)} < T_{\mathrm{WOV}(n-1)}$) such that the last instant of the new shorter $T_{\mathrm{WOV}(n)}$ is early enough that data arrive at the input of the nth latch just before the end of its critical transparency.

This algorithm is summarized in Table I. For simplicity of discussion, it is assumed that the cycle time of operation is long enough to make intralatch data and clock delays insignificant. In addition, the analysis assumes that the design in question consists of a simple sequence of latches and delays. The more realistic case is that of a latch topology in which the path of data is governed by logical constraints and multiple paths to sequentials holding machine state exist.

The basic logic of this algorithm is that the last event in the current window of vulnerability has the worst timing margin with respect to functionality of the design at a given frequency. If this event propagates to the input of the next latch in the path during transparency (Case 2), there is a chance that it will subsequently be able to impact the state of the machine, as would a data transition due to an intentional path through the logic to the input of this latch. It is also possible—as for

TABLE I
ITERATIVE ALGORITHM FOR THE CALCULATION OF THE TD OF LATCHES IN A FLOW-THROUGH DATA PATH DESIGN

Case 1: *Last $T_{wov(n-1)}$ event propagates to latch n before critical phase*

$$T_{wov(n-1)} + delay_1 + \cdots + delay_n < (n-1)T_{phase} \Longrightarrow$$

$$T_{wov(final)} = T_{wov(n)} = T_{wov(n-1)}$$

(*done*)

Case 2: *Last $T_{wov(n-1)}$ event propagates to latch n during critical phase*

$$(n-1)T_{phase} \leq T_{wov(n-1)} + delay_1 + \cdots + delay_n \leq nT_{phase} \Longrightarrow$$

$$T_{wov(n)} = T_{wov(n-1)}$$

(*continue to next latch*)

Case 3: *Last $T_{wov(n-1)}$ event propagates to latch n after critical phase*

$$T_{wov(n-1)} + delay_1 + \cdots + delay_n > nT_{phase} \Longrightarrow$$

$$T_{wov(n)} = nT_{phase} - (delay_1 + delay_2 + \cdots + delay_n)$$

(*continue to next latch*)

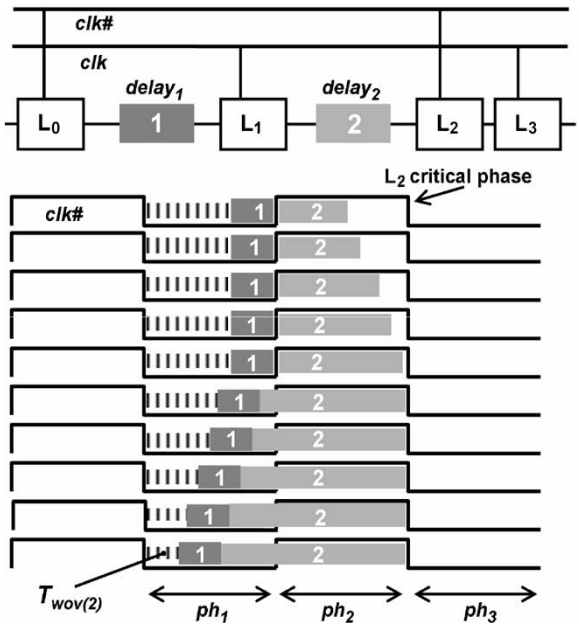

Fig. 11. Graphical interpretation of the TD analysis for latch L_0. Note the impact of delay_1 in reducing the initial value of $T_{\mathrm{WOV}(2)}$ and the constant value of $T_{\mathrm{WOV}(2)}$ until delay_2 becomes large enough to cause SER events late in $T_{\mathrm{WOV}(2)}$ to miss the "critical phase" for latch L_2.

ordinary paths with inadequate timing margin—that subsequent delays in the path will make it impossible for this event to impact machine state. This means that the T_{WOV} defined by Case 2 in the algorithm is an upper bound on $T_{\mathrm{WOV(final)}}$ and that the analysis must continue to determine the final window of vulnerability. In Case 3, the last soft radiation event propagates to the input of the next latch in the chain "after" the critical transparency of this latch. This indicates that delays in the propagation path have made it impossible for some of the later events in the current window of vulnerability to impact machine state. Therefore, the T_{WOV} defined by Case 3 reflects a new upper bound for $T_{\mathrm{WOV(final)}}$. The new upper bound on the window of vulnerability reflects a narrower window of vulnerability, one during which all events again have the possibility of impacting machine state. Case 1 is the condition in which the latest event in the current window of vulnerability propagates to the next latch before its critical transparency. Since any transition occurring before a latch's critical transparency will have a timing margin equal to or greater than the worst designed path through the latch, all such events are guaranteed by the data path design to have adequate timing margin to impact machine state. This implies that there is no potential for further reduction of the window of vulnerability, so the current T_{WOV} is $T_{\mathrm{WOV(final)}}$. As the algorithm executes, the upper bound for T_{WOV} is reduced monotonically until an interlatch delay is encountered that triggers Case 1.

Fig. 11 shows a case that demonstrates the algorithm. The topology is a sequence of latches (L_0 and L_1) followed by delay_1 and delay_2, respectively, and terminating for simplicity in a MSFF consisting of latches L_2 and L_3. Soft radiation events are assumed to impact L_0 during phase ph_1. The initial window of vulnerability $(T_{\mathrm{WOV}(1)})$ for latch L_0 is simply $T_{\mathrm{phase}} - \mathrm{delay}_1$. This is because any radiation event that ultimately impacts the machine state must propagate to the input of L_1 before L_1 becomes nontransparent at the end of ph_1. In the nomenclature of this discussion, ph_1 is the critical phase for L_1 for SER events in L_0. As discussed above, the fact that the last instant of $T_{\mathrm{WOV}(1)}$ generates a transition during the transparency of L_1 means that $T_{\mathrm{WOV}(1)}$ is only an upper bound on $T_{\mathrm{WOV(final)}}$ since subsequent delays could further reduce the window of vulnerability. This is illustrated by the impact that delay_2 has on the window of vulnerability. Once delay_2 becomes larger than a phase, the delay it adds causes radiation events propagating from late in $T_{\mathrm{WOV}(1)}$ to miss transparency at L_2. So $T_{\mathrm{WOV}(2)} < T_{\mathrm{WOV}(1)}$. In this case, the delay between L_2 and L_3 is zero (consistent with the identification of L_2 and L_3 as latches in a MSFF). This means that to first order all SER events in $T_{\mathrm{WOV}(2)}$ will cause the input of L_3 to toggle before the beginning of its critical phase (ph_3), and this in turn means that all of these events have sufficient timing margin to impact latches holding machine state. This is the "case 1" condition that terminates the algorithm, giving $T_{\mathrm{WOV}(2)} = T_{\mathrm{WOV(final)}}$ and $\mathrm{TD} = T_{\mathrm{WOV(final)}}/T_{\mathrm{cycle}}$.

Fig. 12 shows the implementation of the algorithm for a four-latch structure, graphically demonstrating the distinction between case 2 and case 3 (propagation to the input of the L_3 latch during and after the critical phase, respectively) and how T_{WOV} is determined in each case. Fig. 13 shows a comparison between the results predicted by the algorithm and the results of a transistor-level simulation of the TD factor [16] for various values of delay_3 for this topology. The close correspondence between the algorithm's prediction and the simulated result is due to the use of a very long cycle time in the simulations. This reduces the TD factors introduced by intralatch clock and data delays that are not included in the algorithm.

Fig. 14 shows a graphical analysis of the TD for six latches in a sequence terminated by an MSFF. In this example, the

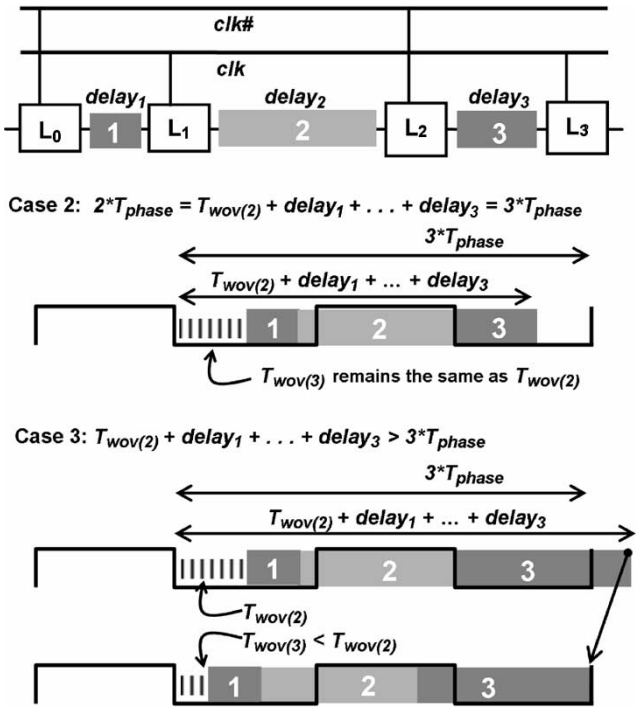

Fig. 12. Example of two possibilities for the addition of the third delay to the TD analysis of a latch (L_0) in a generic flow-through latch-based data path illustrating how the algorithm would be used to calculate $T_{\mathrm{WOV}(3)}$. Case 2: the last SER event in $T_{\mathrm{WOV}(2)}$ arrives during the critical phase, so $T_{\mathrm{WOV}(3)} = T_{\mathrm{WOV}(2)}$. Case 3: the last SER event in $T_{\mathrm{WOV}(2)}$ "arrives after" the critical phase, necessitating a new reduced value for $T_{\mathrm{WOV}(3)}$.

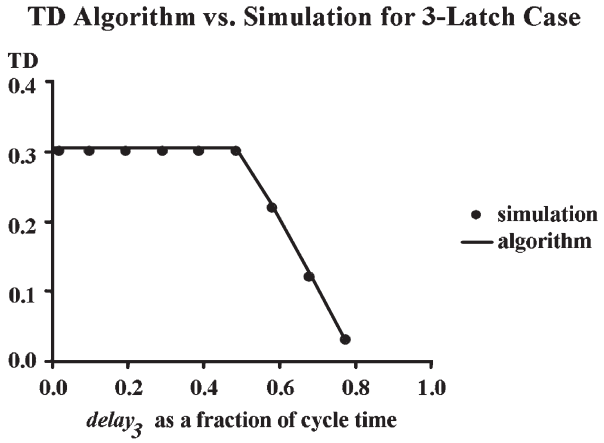

Fig. 13. Algorithmic and simulated TD for L_0 in Fig. 12.

delays between latch L_0 (the one impacted by the SER events) and the MSFF are such that T_{WOV} is not diminished by critical transparency constraints for the first five latches nor are the delays short enough to cause the analysis to terminate before extending to the MSFF. This case suggests that the determination of the overall TD for all the latches in a latch-based design is computationally intensive. However, analysis of typical distributions of interlatch delays suggests that the net TD for a collection of latch-based paths can be determined to adequate accuracy by limiting the analysis to a depth of two or three delays. Fig. 15 shows the average TD calculated using the algorithm described above for a set of randomly assigned delays in the range of 0.1 to 0.6 phases evaluated for path depths

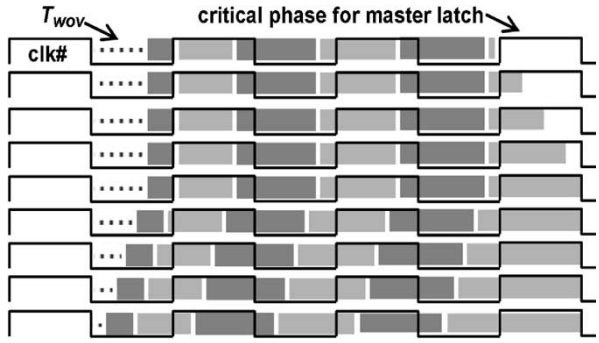

Fig. 14. Graphical solution to the TD for the first of six latches in a path terminated by an MSFF.

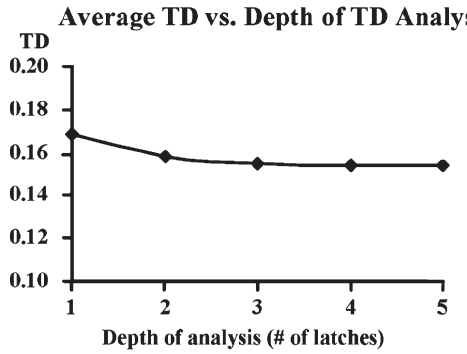

Fig. 15. Average TD calculated using the algorithm described above for a set of randomly assigned delays in the range of 0.1–0.6 phases evaluated for path depths between one and five phases.

between one and five phases. Note how little difference exists in the average TD result between the value for a two-phase analysis and that for a five-phase analysis.

C. TD of Other Sequential Elements

Pulse latches can be treated as FFs if the pulse width is small compared to the cycle time. From a TD modeling point of view, (2) accurately reflects the TD of individual pulse latches in all cases, i.e., independent of the pulse width. The only difference is that for pulse latches the multiplication factor in (2) is close to 1 and not 0.5 as in the latch and FF cases. The exact value depends on the pulse width ($T_{\mathrm{pulse\ width}}$) and can be approximated by

$$\frac{(T_{\mathrm{cycle}} - T_{\mathrm{pulse\ width}})}{T_{\mathrm{cycle}}}. \tag{3}$$

The discussion above is valid only for free running clocks. It is important to note that for gated clocks the TD factor is independent of propagation time. There are two types of clock gating that needs to be taken into account.

1) Inactivity—the circuit/block/unit is not in use and hence clock gating would save power. Example, FP unit while performing integer arithmetic.
2) Store data for next N clock cycle(s). Example, waiting for a stalled condition to clear.

Case (1) is covered by LD since the clock is gated when the block is not in use. TD should thus be applied in the same

way as for a free running clock. Case (2) is not covered by LD and should be addressed differently. This is typically the case when the sequential holds a value (i.e., is not being written), but the logic downstream reads the state stored in the sequential. During clock gating, this sequential therefore behaves very similar to a memory cell and its TD equals 1. When the clock is running, TD depends on the propagation time and is accurately described by (2), (3), and the algorithm in Table I.

D. Average TD for Sequentials

Until now, we have discussed how to assess the TD of various types of sequentials. What one really is interested in is a conservative estimate of the average TD of a product. The calculation of the average product TD factor for sequentials involves two steps:

1) modeling the dependence of TDs on the propagation delay in the combinational logic at use conditions (i.e., for given Vcc, temperature, clock speed, etc.) according to (2) and (3) and the algorithm in Table I;
2) extracting the chip-level distribution of propagation delays.

The average TD of a circuit block can be estimated by integrating the sequential TD for a fixed $\Delta t_{\rm prop}$ over the corresponding delay distributions (one for FFs, one for latches, etc.). Using min delay statistics for the delay distribution to bound the average TD conservatively, we get TD values of the order of 30% or less for a typical microprocessor.

E. TD of Static Combinational Logic

A glitch induced by ionizing radiation needs to arrive at the receiving sequential within the setup $(\Delta t_{\rm setup})$ and hold time $(\Delta t_{\rm H})$ window in order to result in a latched fault. Due to the attenuating nature of glitch propagation, TD is impacted by the speed and number of gates between the node/device where the glitch has been induced and the receiver. According to Nguyen and Yagil [15], the TD of static logic for a single path can be approximated by

$$\mathrm{TD}_{\rm static} \sim \frac{\Delta t_{\rm setup} + \Delta t_{\rm H} + \Delta t_{\rm glitch}}{T_{\rm cycle}} \tag{4}$$

where $\Delta t_{\rm glitch}$ equals the width of the glitch at the receiver. Equation (4) predicts that the SER of static logic should increase linearly with clock speed. This is indeed observed in our SPICE-based SER simulations shown in Fig. 16. The simulation procedure has been described in detail in [17]. Fig. 16 depicts the SER trend of an 8-bit static adder as a function of clock frequency.

Fig. 17 shows the SER as a function of injection time for various adder nodes. The width of the SER response divided by the cycle time equals the TD of the corresponding node. The TD values of the examples shown in the figure vary from about 0.12 to 0.24. Equation (4) predicts TD of the order of 0.2, which is in good accord with these values. Note that the pulses shown in the figure peak at slightly different times. This is due to the fact that the selected nodes are located at different gate

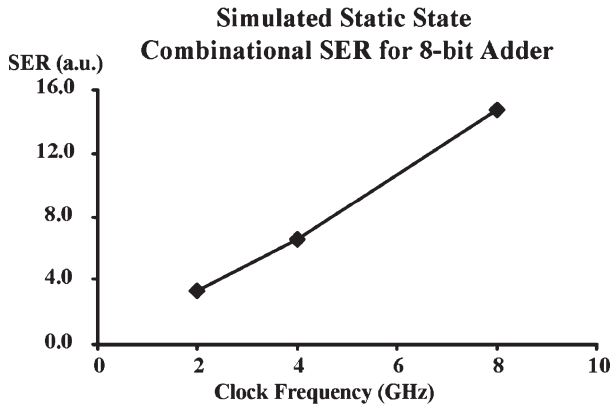

Fig. 16. Simulated static combinational SER of an 8-bit adder plotted as a function of clock speed.

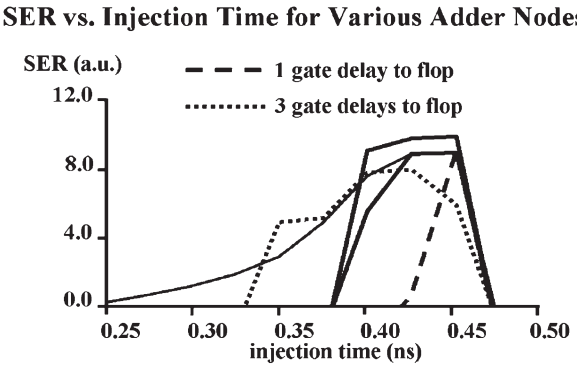

Fig. 17. Simulated SER as a function of upset injection time for various nodes in static adder.

delays from the receiving sequentials. In case of the narrowest response (centered at 0.45 ns, dashed in the diagram), the node is only one stage away, whereas the widest response (dotted in the diagram) is three stages from the receiving sequential. The farther a node is away, the sooner the glitch needs to be induced to arrive within the setup and hold time window of the receiving sequential.

The TD of static combinational logic is difficult to extract given that tool runs have to operate on the chip-level and are taxed by the need to run circuit simulations for every path and node. In that case, we suggest to use $(\Delta t_{\rm setup} + \Delta t_{\rm H})$ as a conservative estimate of the glitch width leading to a conservative TD estimate of $2(\Delta t_{\rm setup} + \Delta t_{\rm H})/T_{\rm cycle}$ for static logic.

One consequence of the different clock frequency dependencies of the TD factors of static combinational logic and of sequentials is that for critical (i.e., long) data paths, the static combinational SER can be of the same order of magnitude as the SER of sequentials. This is shown in Fig. 18.

The static logic located in a path that feeds into the dynamic precharged logic is characterized by different TD factors than static logic feeding into sequentials. The main reason is that for precharged logic only one edge (rising or falling depending on the implementation) is relevant and once the right edge has tripped the precharged logic no recovery is possible. The width of the pulse has to be sufficient to cause an incorrect evaluation, but the consequence impacts the nominal SER and not TD.

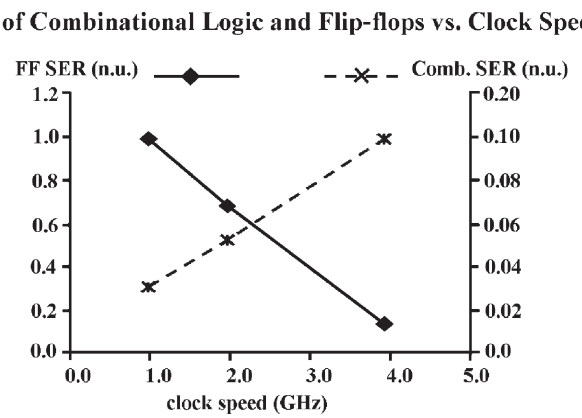

Fig. 18. Simulated static combinational SER and FF SER in a long data path are plotted as a function clock speed. One can clearly observe the different clock frequency trends.

If the precharged logic evaluates only for $0 \rightarrow 1$ inputs, then $1 \rightarrow 0$ upsets cannot be latched at all and the corresponding $\mathrm{TD} = 0$.

F. TD of Dynamic Combinational Logic

Typically, dynamic or domino logic consists of flow-through latches separated by dynamic combinational gates. In this configuration, the latches hold data for a phase while the dynamic nodes of the subsequent gates are pulled out of their precharge state (or remain in their precharge state) as data cascade from one gate to the next. Since the delay between latches is (by design) a phase or less, the delays between subsequent latches in the chain do not impact the latch's window of vulnerability and the time derating of a latch in this structure is simply determined by the delay to the next latch in the chain. So its window of vulnerability (T_{WOV}) is $T_{\mathrm{phase}} - \mathrm{delay}_c$, where delay_c is the total delay of the combinational logic to the next latch and $\mathrm{TD} = T_{\mathrm{WOV}}/T_{\mathrm{cycle}}$. The window of vulnerability for the combinational dynamic gates between the latches is determined in the same way: T_{WOV} is $T_{\mathrm{phase}} - \mathrm{delay}_c$, where delay_c is the total delay of the combinational logic from the given dynamic gate to the next latch.

VII. LD Estimation

The concept of LD arises from the observation that not all SEUs result in a machine failure. LD is a measure of how the device logically reacts to a strike. It is dependent on the applications (how the device is utilized) and the microarchitecture of the design (how the device responds to an error). LD for a particular unit or functional block can be as high as 1 (no derating) when every strike is fatal (strike in a control register that changes the behavior of the machine) or as low as zero (100% derating) when the strike is benign (strike in a branch prediction structure that only causes a momentary performance glitch but no functional failure). Mukherjee *et al.* [28], [29] used the terminology architectural vulnerability factor (AVF) when referred to LD. The smaller the LD, the smaller is the AVF and vice versa.

Mukherjee *et al.* [28] described three methods that can be used to compute or estimate the LD of different processor structures: 1) analytical models using Little's law [30], 2) architecturally correct execution (ACE) analysis using performance models (simulators), and 3) statistical fault injection (SFI).

A. LD Estimate Using Little's Law

As described in [28], there are cases when data flow unmodified and without duplication through a structure. Little's law [30] can be used to estimate the LD of such a structure before a performance model (high-level description of the important machine components) or register transfer language (RTL) model is available. Biswas *et al.* [31] introduced the concept of ACE as any execution that generates correct results as observed by a user. Applying Little's law, one can compute the ACE bits of a structure as the product of the average bandwidth of ACE bits entering the structure and their residency in the structure. LD can then be estimated as the ratio of the ACE bits to the total bits in the structure. This method can be used to estimate LDs for structures commonly found in a processor including Instruction Queue, Bus Queue, Cache Queue, etc.

B. LD Estimate Using ACE Analysis

When the performance models are available, they can also be used to estimate the LDs of various structures in the machine using lifetime analysis to identify the fraction of time the bits flowing through the structure are ACE. The fraction of time a bit contains an ACE state is the LD of the bit or structure. Please refer to [28] for a complete treatment on using the analytical model and performance model to derive LDs of some sample structures found in a typical processor.

Using Little's law and ACE analysis to estimate LDs in the early design stage can be effective. These techniques, however, require detailed knowledge and understanding of the design to be able to capture the LD information appropriately. Depending on the complexity of the design, this method may not always be the most straightforward approach and may lead to unintended inaccuracies.

C. SFI

SFI is a technique that can be used to derive the LDs of any component in the chip once the chip RTL or schematic is available. SFI introduces a bit flip into the model at a particular time. Simulations are then run forward from the fault injection point for a predetermined number of clock cycles and the states of the machine are compared with the error-free machine states at the end of the simulation. As discussed earlier, the machine states include the architecture and memory states as well as the states represented by the chip's outputs. If the comparison does not result in a mismatch, the error is either latent in the processor and is not visible in the observed states, or has been masked and disappeared. Although one bit flip can result in many mismatches in the observed states, each simulation run will result in either a pass (no mismatches) or fail (one or more mismatches). The LD of the structure being studied is estimated as the ratio of the total fails divided by the total number of bit flips.

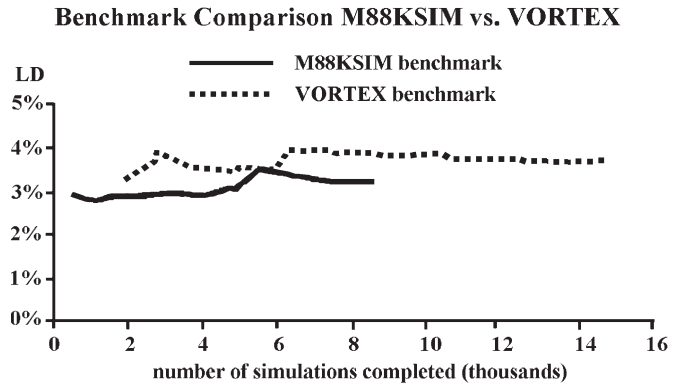

Fig. 19. LD versus number of completed simulations for M88KSIM and VORTEX benchmarks.

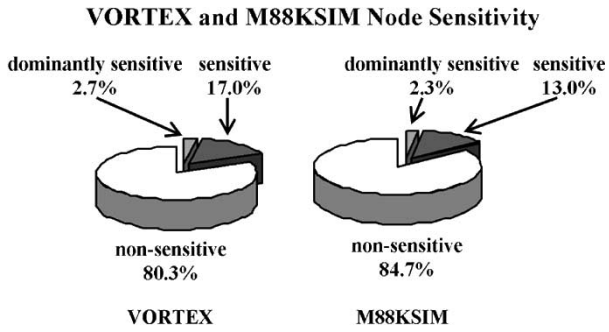

Fig. 20. VORTEX and M88KSIM node sensitivity.

An SFI experiment was performed on an Itanium-class processor to derive the LD for latches in the processor. Given the total number of latches and the number of points in time when each latch could flip during the applications, there were 150 trillion flips that could happen if each latch was flipped on every cycle of simulation. This of course was not achievable, thus sampling in both space (which latch to flip) and time (when to flip) was necessary.

Equation (5) was used to determine the minimum number of latches required for sampling, given a desired accuracy and confidence interval when the number of latches and initial LD estimate are known. Using (5) with N_i of 150 K latches, p_i of 0.5, z of 1.64, and a 95% accuracy, the number of latches required for sampling was 300. The 300 latches were uniformly randomly selected from all the latches that exist in the RTL, i.e.,

$$n_i = \frac{N_i}{1 + \left(\frac{A^2(N_i - 1)}{4z^2 p_i q_i}\right)} \tag{5}$$

where

- N_i number of population or number of total latches;
- n_i number of latch sampled;
- p_i derating factor estimate;
- q_i $1 - p_i$;
- z 1.64 for 90% confidence level two-sided confidence interval;
- A accuracy $= 95\%$.

Time sampling was also necessary. We chose 100 uniform sampling points across each application as the time to inject the fault. For each latch chosen to receive an SER strike, 100 different time instances were chosen as the time to flip the latch, resulting in a total of 30 K RTL simulations. The 100 time sampling points were chosen somewhat arbitrarily and more sample points would provide a more accurate LD estimate. We used two SpecINT95 benchmarks M88KSIM and VORTEX in this SFI study. LD results are shown in Figs. 19 and 20.

Fig. 19 from [19] shows the results of the LD for latches using M88KSIM and VORTEX benchmarks. The SFI experiment shows an LD of 3.3% for M88KSIM and 3.8% for VORTEX. Fig. 20 shows the latch node sensitivity. Nodes can be grouped into three categories: dominantly sensitive (95%–100% failure rate), sensitive (0.1%–94% failure rate), and nonsensitive (0% failure rate). 2%–3% of the nodes selected are very susceptible to faults regardless of the code stream; 80%–85% are insensitive to any type of the faults within the limited code stream sampled; and 13%–17% are sensitive to the faults. Results also show that among the dominantly sensitive node category, both VORTEX and M88KSIM show a similar set of failure nodes. These nodes are the architectural registers and the global control signals such as reset, flush, and stall signals. The architectural state nodes dominate the dominantly sensitive category due to their "stickiness" or long lifetime within the program execution.

A similar fault injection study was reported by Kim *et al.* [32] when they performed the fault injection study at the RTL level on the Sun Microsystems picoJava-II microprocessor. They used the SES as a metric, defined as the probability that a soft error causes the processor to enter an incorrect architectural state. Similar to our study, fault injection was also performed by sampling both space and time domains. Similar to our findings, they reported that there were very few sensitive fault locations. Others [33]–[38] also performed fault injection analyses using various designs and methods.

In summary, each of the above three methods has its own advantages and disadvantages; the methods complement one other. It is possible to combine all three techniques to judiciously produce the desired LDs for a given design. For instance, Little's law and ACE can be used to derive LDs for structures that are well understood like caches, instruction queue, and register files. For random latches, it is more effective to use SFI since it is very difficult to identify the ACE and un-ACE property for each random latch in the design.

D. LD for Combinational Logic

As technology progresses with further shrinking of the clock period, FIT contribution due to combinational logic is also increasing and thus cannot be ignored. It is therefore important to be able to derive the LD of combinational logic. The effort needed to derive accurate LD per element including sequentials and combinational logic is rather costly hence not practical for large designs. A compromise is to inject SEU only to sequential elements (latches) to derive LD per latch. These LDs can then be used to estimate the LD of combinational elements between those data path latches. The idea is based on the fact that for a synchronous design, combinational logic connects one set of latches to another. Therefore, the data path latches connected to

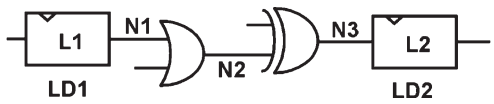

Fig. 21. LD of combinational gates.

combinatorial logic bound the LD of each combinatorial gate in the data path.

LD_1 in Fig. 21 includes the logic masking from latch L_1 to latch L_2 as well as the LD of latch L_2. Therefore, in the above example, $LD_1 < LD_2$, meaning Latch 1 has more LD than Latch 2. Similarly, $LD_1 \leq LD(N_2) \leq LD(N_3) \leq LD_2$. Thus, one could apply some heuristics to estimate the LD of combinatorial nodes based on the LDs of the surrounding latches.

The concept that data path latches bound the LDs of combinational logic in the data path does not necessarily hold true for logic in the control path. Logic in the control path may be bounded only by the leading latch (shown as L_1 in Fig. 21) and not by L_2. In these cases, the LD of the combinational logic will approach the LD of the leading latch and can be estimated to be the same as LD_1.

A coarse estimate is to inject SEU only on selected latches and collect statistics on the average LD per block (say a unit). This technique means random sampling in both time and space (space means selecting random latches within the block). The outcome is a single LD for all nodes in the block, both sequential and combinational. Recalling that combinational LD in the data path is bounded by the LDs of the latches connected to it, combinational LD has the same value as that of the latches if all latches have the same LD. Considering the huge number of latches and combinational logic within a CPU, and the large number of different applications that can run on it, this later approach is the most practical one. The key observation here is that a combinational LD can be derived from latch LD with sufficient accuracy for FIT estimation purposes.

E. LD and Fine Grain Clock Gating

Designs that utilize fine grain clock gating have built-in LD data: the clock gating is actually a measure of circuit utilization and hence of its LD. That is, if the clock is only activated when the circuit is actually used under a specific application, then its LD when the clock is active is "1" (no derating) and "0" (full derating) otherwise. Thus, the duty cycle of the clock gating control signal is the LD of the circuit. In this case, LD can be estimated based on clock gating statistics, as most are already available from power analysis simulations. However, one still needs to perform benchmark simulations and generate statistics of the duty cycle of all clock-gating signals based on the typical behavior of the chip.

VIII. Chip-Level FIT Estimation

A. FIT Estimation in Early in the Design Effort

Early FIT estimation is based on architectural considerations and models as well as on a target library. Estimating nominal FIT rates of library cells enables full chip FIT estimates based on early estimates of gate count, gate sizes, average fan-out, and usage conditions (temperature and voltage). For sequential cells, one needs to consider not only the storage nodes (or recycled loops) but also the control signals and data path buffers when estimating a cell's FIT rate. These nodes are later attributed to "storage," "control logic," and "random logic" by the SER tools but need to be included in the nominal FIT of standard cells in order to enable realistic early FIT estimation. Domino and static gates should be connected to default loads before nominal FIT estimation takes place. Typical fan-out is a good method for load estimation. Path information is not available at an early stage but default criterion for static gates can be used for some estimates. Typical setup and hold times of the latches in the cell library are the best choice for W_{crit}. V_{trip} can simply be set to half the operating voltage.

Libraries normally consist of hundreds or thousands of cells. However, only a small subset of cells is commonly used while the rest are not used or have few instances per design. Moreover, SER is sensitive to node capacitance and to transistors' sizes; hence, "strong" gates (large transistors, built to drive large capacitive load) have negligible contribution to total FIT rates. Therefore, it is sufficient to perform nominal FIT analysis only on a subset of library cells.

Armed with nominal FIT of the library cell elements, the total number of cells expected in the design, and the TD and LD for each sequential holding machine state and nonmachine state, the chip FIT rate can be estimated by summing the estimated FIT of all elements. This early spreadsheet can be very useful in determining whether or not the device meets its goals and helps identify areas of focus for FIT reduction. In the early design stage, LD is the most difficult estimate to refine.

B. FIT Estimation in Later Design Stages

Since FIT rate is sensitive to details of the implementation (logic families, gate types), transistors' sizes, capacitive load, and details of the combinatorial logic paths, accurate FIT rates can only be derived once the design is complete. The final FIT estimate often provides the only validation that the product in question meets its FIT spec target.

In order to perform accurate nominal FIT estimation for all circuits in the design, one needs automation that can isolate gates (latches, domino, and static) and perform nominal FIT simulation for each gate. This requires the following capabilities:

- schematics for each design block;
- real or estimated parasitics;
- partition of the design into stages that represent the basic building blocks: diffusion connected networks (DCNs) and loops;
- classification of the stages into predefined primitives, such as latches, domino, and static gates;
- identification of "failing nodes" to be tracked and derive their failure criterion for each stage (recycled node that can flip in case of latch and domino; output node of a static gate as well as W_{crit} and V_{trip} based on the downstream paths);

- perform nominal FIT estimation (simulations) for each stage. This capability requires the scanning of all valid input vectors and automatically identifying the nodes on which SEUs should be injected.

TD could either be done in this stage or at the full-chip roll-up stage. Accurate TD, however, is based on details of the circuit and therefore is best handled in this stage. Path analysis and usage of timing data such as required and valid time windows as well as duty cycle of clocks enable reasonably accurate TD estimation per node.

If detailed LD data are available, namely, LD per node or per latch or based on fine grain clock gating, then LD should be applied in this stage. In particular, heuristics that estimate the LD of the combinatorial logic based on latch LD is best done here since the full context of the schematics is available and can be used for adequate heuristics.

The outcome of this stage of FIT analysis is a report for full-chip roll-up: weighted average of FIT rate per node or per stage, based on SP, and possibly also includes TD and LD data. Another important report of the above analysis is a hierarchical report that maps the FIT rate to the original hierarchies of the schematics. Such reports have twofold advantages: 1) identify high FIT contributors that can be fixed by small effort; and 2) enable detailed comparison between early and late FIT estimations, which is mainly useful for future designs.

The final stage of FIT analysis for a product is the full chip roll-up. Here, one needs to provide the total FIT rate at all relevant categories (detected, undetected, etc.) and its breakdown to specific blocks of the design as well as to circuit types. The roll-up depends on the technique used for LD and TD estimations, which might be included in the nominal derated FIT rates or applied only in this stage.

Machine states (MS): the roll-up consists of nominal FIT rates of the various blocks holding machine state and applies LD. As discussed above, machine state elements tend to be in "storage" mode most of the time, hence no time derating (namely TD $= 1$).

Non-MS: if the nominal FIT rates already include TD and LD, then full chip roll-up is just a summation of the derated nominal FIT of all non-MS blocks in the design. Otherwise, LD and TD are applied in this stage based on average block derating and on default TD per circuit type.

On top of the above, the full-chip roll-up stage is responsible for the classification of SER contributions to DUE and SDC according to the role of each block in the design. Further LD is also applied where applicable. For example, branch prediction cache has LD $= 0$ since it will never cause a wrong behavior (just a tiny performance hit). A sample roll-up report of SDC FIT is shown in Table II.

In most cases, the machine state components dominate the FIT rate of most modern processors because of their sheer capacities (caches, TLBs, register files). Most caches of reasonable sizes (8 kB and above) are protected by either parity or ECC. Some TLB RAM arrays of significant sizes are also protected as well as register files. With most if not all machine state components already protected, the nonmachine state components start dominating the FIT rates of the processor. As technology progresses in deeper submicron technology, we saw that TD increases with clock frequency and thus depending on the number of sequentials compared to combinational logic. For some designs starting at the 90-nm technology, the FIT contributions of combinational logic could be significant and approaching those of sequentials.

TABLE II
SAMPLE FULL-CHIP FIT ROLLUP (ARBITRARY NUMBERS)

	Element type	TD	LD	Nominal FIT	Derated FIT
MS	Caches w/o ECC	1	0.3	100	30
	Register Files	1	0.4	50	20
	TLBs	1	0.4	20	8
	ARs, CRs	1	1	5	5
Non-MS	Latches	0.5	0.2	100	10
	Static data-path	0.2	0.2	200	8
	Static Control	0.5	0.2	50	5
	Domino data-path	0.3	0.2	50	3
	Domino control	0.3	0.2	10	0.6
Total					89.6

IX. SUMMARY

In this paper, we described a systematic approach to soft error rate (SER) estimation using a typical high-performance processor as the basis for discussion and using typical circuits found in such a processor for illustrative purposes. We discussed the concepts of nominal failure in time (FIT), timing derating (TD), and logic derating (LD), and provided methods for estimating and simulating them in early and late design stages.

SER estimation of a processor is divided into sequentials holding machine states and nonmachine states, where sequentials holding machine states in this context include caches. Once the caches and other large arrays are SER protected, FIT can be dominated by nonmachine state elements. These include sequentials and static and dynamic combinatorial logic. In order to meet FIT rate specs, one needs both good FIT estimation and FIT reduction techniques. Parity and error correcting code (ECC) are good solutions for arrays. Special layout rules can reduce the sensitivity to multibit errors and let parity and ECC protect single-bit events. Non-array circuits are harder to protect, yet doable with certain area, power, and/or performance penalty [15]. This includes circuits designed to be SER immune, as well as time and space redundancy [39]. The most drastic SER reduction measure is full redundancy, which also provides the lowest FIT rate solution.

ACKNOWLEDGMENT

This paper summarizes work from many contributors within Intel. In particular, the authors thank T.-W. Lee for static glitch propagation analysis, Y. Levani for transient simulations, S. Chakravarty and T. Chandra for fault injection tool development, C. Dai and S. Walstra for SER model definition, and S. Tu for fault injection studies. In addition, the authors thank

J. Maiz, J. Crawford, C. Constantinescu, N. Tam, D. Regenold, and many others for their insights and suggestions. Finally, regards to T. May and M. Woods.

REFERENCES

[1] T. May and M. H. Woods, "Alpha-particle-induced soft error in dynamic memories," *IEEE Trans. Electron Devices*, vol. 26, no. 1, pp. 2–9, Jan. 1979.

[2] J. Ziegler and W. A. Lanford, "Effect of cosmic rays on computer memories," *Science*, vol. 206, no. 4420, p. 776, Nov. 1979.

[3] T. J. O'Gorman *et al.*, "Field testing for cosmic ray soft errors in semiconductor memories," *IBM J. Res. Develop.*, vol. 40, no. 1, p. 41, Jan. 1996.

[4] J. F. Ziegler *et al.*, "Accelerated testing for cosmic soft-errors rate," *IBM J. Res. Develop.*, vol. 40, no. 1, p. 51, Jan. 1996.

[5] G. R. Srinivasan, "Modeling the cosmic-ray induced soft-error rate in integrated circuits: An overview," *IBM J. Res. Develop.*, vol. 40, no. 1, p. 77, Jan. 1996.

[6] H. H. K. Tang, "Nuclear physics of cosmic ray interaction with semiconductor materials: Particle induced soft errors from a physicist's perspective," *IBM J. Res. Develop.*, vol. 40, no. 1, p. 91, Jan. 1996.

[7] P. C. Murley and G. R. Srinivasan, "Soft error Monte Carlo modeling program, SEMM," *IBM J. Res. Develop.*, vol. 40, no. 1, p. 109, Jan. 1996.

[8] L. B. Freeman, "Critical charge calculations for a bipolar SRAM array," *IBM J. Res. Develop.*, vol. 40, no. 1, p. 119, Jan. 1996.

[9] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, 2002, pp. 329–332.

[10] N. Cohen, T. S. Sriram, N. Leland, D. Moyer, S. Butler, and R. Flatley, "Soft error considerations for deep-submicron CMOS circuit applications," in *Int. Electron Devices Meeting (IEDM)*, Washington, DC, 1999, pp. 315–318.

[11] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. Int. Conf. Dependable Systems and Networks*, Washington, DC, 2002, pp. 389–398.

[12] G. R. Srinivasan, P. C. Murley, and H. K. Tang, "Accurate, predictive modeling of soft error rate due to cosmic rays and chip alpha radiation," in *Proc. Int. Reliability Physics Symp.*, San Jose, CA, 1994, pp. 12–16.

[13] R. J. McPartland, "Circuit simulations of alpha-particle-induced soft errors in MOS dynamic RAMs," *IEEE J. Solid-State Circuits*, vol. 16, no. 1, pp. 31–34, Feb. 1981.

[14] C. Dai, N. Hakim, S. Hareland, J. Maiz, and S.-W. Lee, "Alpha-SER modeling and simulation for sub-0.25 μm CMOS technology," in *Symp. VLSI Technology*, Kyoto, Japan, 1999, pp. 81–82.

[15] H. T. Nguyen and Y. Yagil, "A systematic approach to SER estimation and solutions," in *Proc. 41st Int. Reliability Physics Symp. (IRPS), IEEE EDS*, Dallas, TX, Apr. 2003, pp. 60–70.

[16] N. Seifert and N. Tam, "Timing vulnerability factors of sequentials," *IEEE Trans. Device Mater. Rel.*, vol. 4, no. 3, pp. 516–522, Sep. 2004.

[17] N. Seifert, X. Zhu, and L. Massengill, "Impact of scaling on soft error rates in commercial microprocessors," *IEEE Trans. Nucl. Sci.*, vol. 49, no. 5, pp. 3100–3106, Dec. 2002.

[18] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[19] H. Nguyen *et al.*, "SER logic fail derating: Estimating product FIT rates," in *Topical Research Conf. Reliability*, Austin, TX, Oct. 25–27, 2004, pp. 5–6.

[20] H. Nguyen, "A kind and gentler guide to soft error rate," in *Reliability Physics Tutorial Notes, Int. Reliability Physics Symp.*, Dallas, TX, 2002, p. 121_08.1.

[21] D. C. Bossen, "CMOS soft errors and server design," in *Reliability Physics Tutorial Notes, Int. Reliability Physics Symp.*, Dallas, TX, 2002, pp. 121_07.1–121_07.6.

[22] N. Seifert, V. Ambrose, P. Shipley, M. Pant, and B. Gill, "Radiation induced clock jitter and race," in *Int. Reliability Physics Symp. (IRPS)*, San Jose, CA, Apr. 2005, pp. 215–222.

[23] T. Karnik, B. Bloechel, K. Soumyanath, V. De, and S. Borkar, "Scaling trends of cosmic rays induced soft errors in static latches beyond 0.18 μ," in *Symp. VLSI Circuits Dig. Tech. Papers*, Kyoto, Japan, 2001, pp. 61–62.

[24] S. Hareland, J. Maiz, M. Alavi, K. Mistry, S. Walstra, and C. Dai, "Impact of CMOS scaling and SOI on soft error rates of logic processes," in *VLSI Technology Dig. Tech. Papers*, Kyoto, Japan, 2001, pp. 73–74.

[25] P. Hazucha, T. Karnik, J. Maiz, S. Walstra, B. Bloechel, J. Tschanz, G. Dermer, S. Hareland, P. Armstrong, and S. Borkar, "Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-/spl μm to 90-nm generation," in *Int. Electron Devices Meeting (IEDM)*, Washington, DC, 2003, pp. 21.5.1–21.5.4.

[26] P. E. Dodd, M. R. Shaneyfelt, J. R. Schwank, and G. L. Hash, "Neutron-induced soft errors, latchup, and comparison of SER test methods for SRAM technologies," in *Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, 2002, pp. 333–336.

[27] E. H. Cannon, "SER evaluation and modeling," in *7th Annu. Topical Research Conf. Reliability*, Austin, TX, Oct. 25–27, 2004, p. 6.

[28] S. Mukherjee *et al.*, "The soft error problem: An architectural perspective," in *11th Int. Symp. High Performance Computer Architecture (HPCA)*, San Francisco, CA, 2005, pp. 243–247.

[29] S. S. Mukherjee, C. T. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *36th Annu. Int. Symp. Microarchitecture (ICRO)*, San Diego, CA, Dec. 2003, pp. 24–40.

[30] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[31] A. Biswas, P. Racunas, R. Cheversan, J. Emer, S. S. Mukherjee, and R. Rangan, "Computing the architectural vulnerability factor of address-based structures," in *Int. Symp. Computer Architecture (ISCA)*, Madison, WI, Jun. 2005, pp. 532–543.

[32] S. Kim and A. K. Somani, "Soft error sensitivity characterization for microprocessor dependability enhancement strategy," in *Proc. Int. Conf. Dependable Systems and Networks (DSN)*, Washington, DC, 2002, pp. 416–425.

[33] J. Clark and D. Pradhan, "Fault injection: A method for validating computer-system dependability," *IEEE Computer*, vol. 28, no. 6, pp. 47–56, Jun. 1995.

[34] M. Hsueh, T. Tsai, and R. Iyer, "Fault injection techniques and tools," *IEEE Computer*, vol. 30, no. 4, pp. 75–82, Apr. 1997.

[35] S. Chau, "Fault injection boundary scan design for verification of fault tolerant systems," in *Proc. Int. Test Conf. (ITC)*, Washington, DC, 1994, pp. 667–682.

[36] E. Jenn *et al.*, "Fault injection into VHDL models: The MEFISTO tool," in *Proc. Fault-Tolerant Computing Symp. (FTCS-24)*, Austin, TX, 1994, pp. 66–75.

[37] A. Amendola *et al.*, "Fault behavior observation of a microprocessor system through a VHDL simulation-based fault injection experiment," in *Proc. Conf. EURO Design Automation*, Geneva, Switzerland, 1996, pp. 536–541.

[38] D. Gil *et al.*, "Fault injection into VHDL models: Analysis of the error syndrome of a microcomputer system," in *Proc. Fault-Tolerant Computing Symp. (FTCS-28)*, Vesteras, Sweden, 1998, pp. 418–424, EUROMICRO. [Online]. Available: http://www.informatik.uni-trier.de/~ley/db/conf/euromicro/index.html

[39] S. S. Mukherjee, M. Kontz, and S. K. Reinhardt, "Detailed design and implementation of redundant multithreading alternatives," in *Proc. 29th Annu. Int. Symp. Computer Architecture (ISCA)*, Anchorage, AK, May 2002, pp. 99–110.

Hang T. Nguyen (S'81–M'88) received the B.S.E.E. degree from the University of Tennessee, Knoxville, in 1982 and the M.S.E.E. degree from the Georgia Institute of Technology, Atlanta, in 1984.

She was with General Electric Semiconductor from 1984 to 1987 and has been with Intel since 1987. She has worked on several areas of integrated circuit design, including I/O and memory design as well as CPU architecture. She became an Intel Principal Engineer in 2001 and is currently the Lead CPU Architect for Intel's Consumer Electronics Group. Her areas of interest include SER, on-die interconnects, and low power technologies.

Yoad Yagil received the B.Sc. degree in physics and computer sciences from Bar Illan University, Ramat-Gan, Israel, in 1983, and the M.Sc. and Ph.D. degrees in physics from Tel Aviv University, Tel Aviv, Israel, in 1987 and 1992, respectively.

From 1983 to 1993, he was with Lipman Electronics, Israel, focusing on digital and analog design of embedded microprocessor systems. From 1993 to 1995, he did his post-doc research at Cambridge University, Cambridge, U.K. His research included electrical and optical response of inhomogeneous thin films and electrical and optical spectroscopy of superconductors. He joined Intel in 1995 and became an Intel Principal Engineer in 2000. At Intel, he has worked on various computer-aided design (CAD) tools and technologies, with main focus on interconnect reliability, SER, parasitic extraction, power delivery, and power estimation. He is currently leading the power and circuit simulation group in Intel's CAD organization.

Norbert Seifert (M'00–SM'04) received the M.S. degree in physics from Vanderbilt University, Nashville, TN, and the Ph.D. degree in physics from the Technical University of Vienna, Austria, in 1993. His Ph.D. dissertation focused on radiation-induced defect formation and diffusion in wideband gap ionic crystals.

He has conducted research in a wide range of physics topics, from charge-transfer processes in atomic collisions as a Postdoctoral Associate at the North Carolina State University from 1993 to 1994, to computational fluid dynamics of high-power laser material processing as a Postdoctoral Associate at the Technical University of Vienna from 1994 to 1997. In 1997, he joined the Alpha Development Group (DEC/Compaq/HP), where he worked in the fields of device physics, device reliability, and digital design. He is currently a Staff Reliability and Design Engineer with Intel Corporation, Hillsboro, OR, where he is responsible for SER model development and verification.

Dr. Seifert has served as a Committee Member and Technical Session Chair for IEEE International Reliability Physics Symposium (IRPS) and International On-Line Testing Symposium (IOLTS), and is a frequent Reviewer for IEEE TRANSACTIONS ON DEVICE AND MATERIALS RELIABILITY.

Mike Reitsma received the B.S.E.E. degree from the University of Minnesota, Minneapolis, in 1977.

In 1977, he began work with Intel and has held technical and managerial positions in the design of nonvolatile memories and microprocessors. He is currently a Principal Engineer working in Intel's Enterprise Platforms Group.