Virtually-Aged Sampling DMR: Unifying Circuit Failure Prediction and Circuit Failure Detection

Raghuraman Balasubramanian
University of Wisconsin-Madison
ragh@cs.wisc.edu

Karthikeyan Sankaralingam
University of Wisconsin-Madison
karu@cs.wisc.edu

ABSTRACT

Hardware failure due to wearout is a growing concern. Circuit failure prediction is an approach that is effective if it meets the following requirements: low design complexity, low overheads, generality (supporting various types of wearout including soft and hard breakdown) and high accuracy. State-of-the-art techniques, which typically detect and measure low level circuit properties like gate delay cannot deliver on all four requirements. Moving away from the paradigm of measuring circuit delays is key to satisfying the four design requirements. Our insight is to **virtually age** the processor and thus manifest a wearout fault early – we convert the delay degradation into a logic fault; **expose the fault** and then **detect the fault**. To virtually age the processor, reducing supply voltage effectively mirrors wearout. For fault exposure, we observe that faults in critical paths are naturally exposed and we develop a technique to expose faults along the non-critical paths using clock phase shifting logic. Our system, Aged-SDMR, combines these two mechanisms to expose wearout faults early and detects them using Sampling DMR. We also develop principles to combine these two mechanisms with any detection technique. We implement a prototype system based on the OpenRISC processor on a Xilinx Zync FPGA. We demonstrate that Aged-SDMR is practical and delivers on all four requirements, has area and energy overheads of 9% and 0.7% respectively, takes at most 0.4 days to detect failure after onset and its early warning window is configurable. More generally, Aged-SDMR provides the capability for low-overhead DMR execution without any missed errors and 100% coverage. It is likely to find broad uses within reliability and elsewhere.

Categories and Subject Descriptors: C.4 [Computer Systems Organization] Performance of Systems — Fault Tolerance; C.0 [Computer Systems Organization] General — System architectures

General Terms: Design, Reliability, Performance

Keywords: Fault tolerance, Permanent Fault, Dual-modular redundancy, Sampling, Reliability

1 Introduction

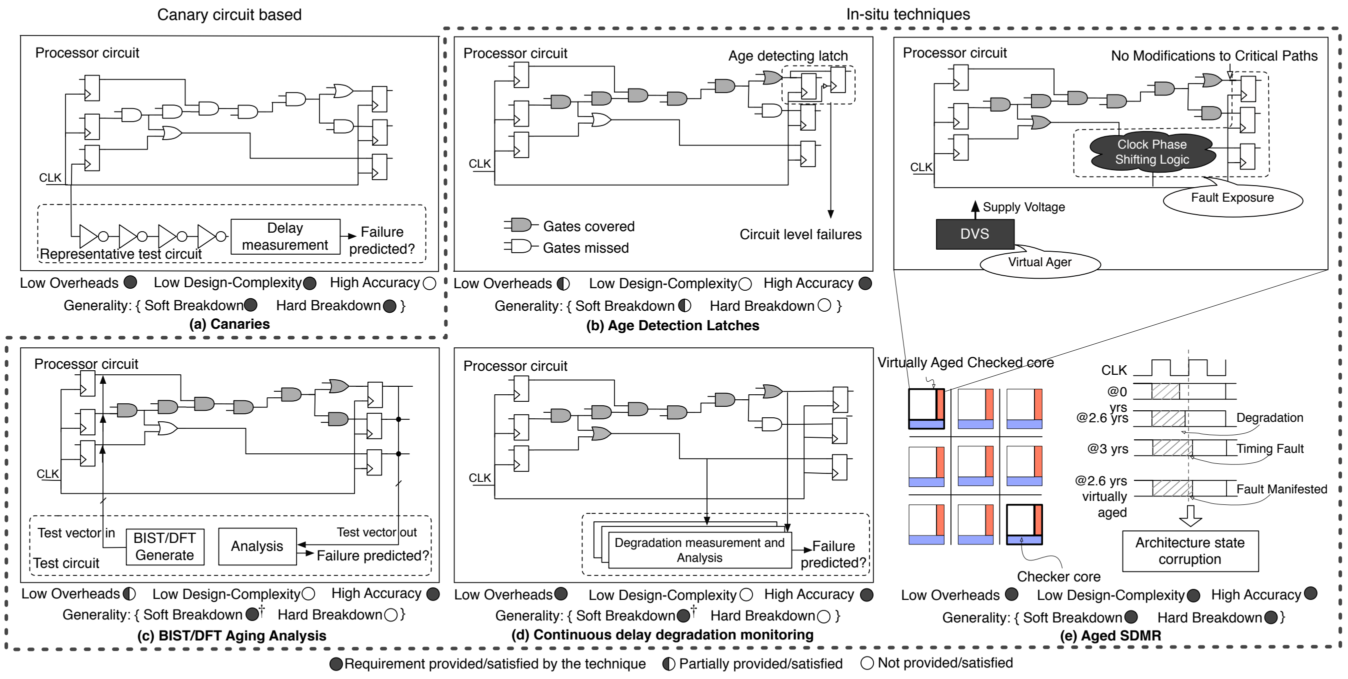
In future generations of silicon technology, microprocessors are increasingly likely to fail in the field due to device faults [2, 8, 10, 31]. One source of the problem is increasing *manufacturing fault escapes* during testing [33]. The other source is faults that appear during the chip's lifetime due to various silicon *aging* or *wearout* phenomenon [23, 52]. The latter is the focus of this paper and can be classified into two broad philosophies: fault detection and fault prediction[1].

Fault detection determines faults as it occurs, whereas fault prediction predicts a fault before it occurs. As outlined in prior works, detection techniques potentially suffer from low fault coverage, long detection latency, silent data corruption and unbounded missed architectural errors [3, 19, 37]. On the other hand, prediction techniques suffer from high overheads, high design complexity, low accuracy and poor generality (i.e. what types of aging mechanisms covered). We define soft and hard breakdowns that characterize wearout and then summarize the representative works in these two paradigms to motivate the principles that can achieve the best of both approaches.

Terms We use the following terms throughout the paper to distinguish two different types of device breakdown. *Soft Breakdown* : As devices wearout, the gates operate slower. When their delays exceed a set period (frequency of operation of the device), they manifest as faults. *Hard Breakdown* : Wearout may cause a catastrophic failure in gates, leading to a permanent fault in its digital logic behavior.

Fault detection Gizopoulos et al. provide a summary and survey of detection techniques [19], some of which we review below. Dual Modular Redundancy (DMR) [20, 28, 35, 46] provides full fault coverage while incurring significant power overheads and design complexity in terms of processor microarchitecture. Diva [5], Argus [34] and BIST-based techniques [13, 47] fall into a paradigm of asymmetric DMR (where the dual is a simplified version of the master) and have lower overheads while they fall short in full fault coverage. SWAT [31] is a software-based low overhead technique, but has an "unbounded" detection latency and does not provide 100% fault coverage. On the other hand, Sampling+DMR [37] provides 100% fault coverage like DMR, while keeping overheads to 1%-5% through sampling, and avoids microarchitecture design complexity. It uses lightweight communication of architectural trace between cores and activates DMR only for small windows at

[1]Complementing and somewhat orthogonal to both is the body of work on repair [4, 22, 26, 53], and recovery [42, 50]


Canary circuit based
In-situ techniques
Processor circuit
CLK
Representative test circuit
Delay measurement
Failure predicted?
Low Overheads ● Low Design-Complexity ● High Accuracy ○
Generality: { Soft Breakdown ● Hard Breakdown ● }
(a) Canaries
Age detecting latch
Gates covered
Gates missed
Circuit level failures
Low Overheads ◐ Low Design-Complexity ○ High Accuracy ●
Generality: { Soft Breakdown ◐ Hard Breakdown ○ }
(b) Age Detection Latches
No Modifications to Critical Paths
Clock Phase Shifting Logic
Fault Exposure
Supply Voltage
DVS
Virtual Ager
Test vector in
BIST/DFT Generate
Analysis
Test vector out
Failure predicted?
Test circuit
Low Overheads ◐ Low Design-Complexity ○ High Accuracy ●
Generality: { Soft Breakdown ●† Hard Breakdown ○ }
(c) BIST/DFT Aging Analysis
Degradation measurement and Analysis
Failure predicted?
Low Overheads ● Low Design-Complexity ○ High Accuracy ●
Generality: { Soft Breakdown ●† Hard Breakdown ○ }
(d) Continuous delay degradation monitoring
Virtually Aged Checked core
Checker core
CLK
@0 yrs
@2.6 yrs
Degradation
@3 yrs
Timing Fault
@2.6 yrs virtually aged
Fault Manifested
Architecture state corruption
Low Overheads ● Low Design-Complexity ● High Accuracy ●
Generality: { Soft Breakdown ● Hard Breakdown ● }
(e) Aged SDMR
● Requirement provided/satisfied by the technique ◐ Partially provided/satisfied ○ Not provided/satisfied

Design complexity means the techniques interferes with the critical path; † : Higher overhead can increase soft breakdown coverage.

Figure 1: Overview of failure prediction techniques and Aged-SDMR

random or periodically chosen intervals. However, from a practical standpoint it suffers from the phenomenon of "missed errors" for device faults that have low architectural error rate. For hardware faults that cause a very low error rate of architectural errors, many sampling windows must occur before the DMR window overlaps with when the error occurs - thus detecting it. In the prior epochs, the architecture errors are missed and caused irrecoverable data corruption. This issue creates a significant problem, without which Sampling-DMR is almost perfect.

Fault prediction Figure 1 graphically demonstrates the underlying mechanism in different prediction techniques. The figure also classifies the techniques according to tradeoffs in *design-complexity*, *overheads*, *generality* and *accuracy*. By generality, we refer to whether the technique can handle various types of wearout, in particular soft *and* hard breakdown.

Broadly, prediction techniques fall into one of two categories, canary-based and in-situ, and in general they are driven by the principle that aging faults start with delay degradation before transitioning into hard breakdown. The main drawback of canary-based techniques (for example [56, 60] - in the interest of space we do not cite the vast literature here) is low *accuracy*, since it is hard to sensitize the canaries similarly to the circuit they are associated with. In-situ techniques attempt to address this problem. Fine-grained techniques [3, 9, 16, 39] operate at the level of each flip-flop and suffer from high *overheads* if applied to all flip-flops. If selectively applied to only the critical paths, they suffer from poor *generality*, as uncovered flip-flops can transition unknowingly to hard breakdown as shown in Figure 1(b). In the figure, we show a missed gate – on a path that has a lot of slack. A delay degradation that this gate incurs will not be reflected as a logic fault. While the gate may transition to a hard breakdown, fine-grained techniques can not predict the fault. Coarse-grained techniques attempt to operate on an entire circuit block to amortize overheads. BIST/DFT-based techniques [18, 49, 62] are shown in Figure 1(c). They suffer from coverage problems in that full coverage for delay/timing faults is hard (in general "model coverage" and fault coverage problems [40, 43]), lack *generality*, and offline measurements miss faults masked by dynamic effects like temperature and noise. Blome et al. [6] propose an online technique similar in spirit - they measure circuit signal delay online (i.e., as regular code runs on the processor) at groups of flip-flops attached to a measurement and analysis block as depicted in Figure 1(d). Although time-sharing this block across several gates mitigates some overhead, it adds to load on critical path. For gathering statistically significant measurements, the gates must be sampled frequently; this puts a limit on the number of gates an analysis block can handle and so is not scalable. Further using several, always on, units to boost coverage has high power and area overhead. Section 3.7 expands more on these issues.

In summary, prior work does not simultaneously provide high accuracy, generality, low overheads and low design complexity. A practical and deployable technique has remained elusive. Building on the prior work, we seek to develop a hardware-only technique (desirable for a microprocessor manufacturer) that can satisfy all four requirements with a paradigm unifying detection and prediction.

1.1 Overview of Aged-SDMR

We observe that the four design requirements — low overheads, low design complexity, high accuracy, and generality — can be achieved by moving away from the paradigm of measuring circuit delays to predict wearout. Our key insight is to *virtually wearout* the processor and thus manifest a wearout fault early — we convert the delay degradation into a logic fault; then we *expose the fault*, and then *detect* the fault — which effectively predicts/detects the wearout. By observing that all wearout faults first start as delay faults, "any" detection technique can be repurposed as a prediction technique. In this work, we specifically repurpose Sampling-DMR since it provides three of four requirements: low overheads, low design complexity and high accuracy. Figure 1(e) shows an overview of our technique called Aged-SDMR. In this section, we first explain how we manifest faults early, followed by two specific mechanisms that together provide generality and then the full system design.

Virtual aging to manifest faults We can force faults to manifest early by virtually degrading (or stressing) the processor being checked. Since soft breakdown affects delays only, reducing V_{dd} (without reducing clock period) will effectively increase the delay of gates, thus manifesting the delay fault. This is denoted by the "Virtual Ager" module in Figure 1(e). By reducing V_{dd} *only during DMR windows*, we effectively create an illusion of the checked processor being *older* than it currently is and ensure that faults always manifest first in the DMR window - eliminating the missed error problem that plagues detection using Sampling+DMR. Further, resetting the voltage level restores the gate back to its current age.

Fault exposure To expose faults, i.e. get circuit logic (or digital) behavior to be affected by the apparent presence of wearout induced by virtual aging, we discover and address two subtle cases to guarantee generality and full coverage. Wearout on gates on the critical path get naturally exposed — gate delay increases and ultimately causes setup-time/hold-time violations or metastability problems that result in a wrong value being captured in a flip-flop. Gates on non-critical paths do not cause any timing faults and if unaddressed, can unknowingly transition to hard breakdown. We observe that a simple clock-phase shifting logic can be added to gates on non-critical paths to effectively expose their delays. Both these cases are indicated in Figure 1(e). Further, only a fraction of gates/paths fall under this category and we develop a low overhead mechanism for generating this phase shifting logic. The timing diagram in Figure 1(e) shows this concept without making the distinction about the type of path. During normal operation, values arrive before the rising clock-edge, and after wearout degradation delays, say in 36 months, the signal arrives after the clock edge. This effect can be manifested and the timing fault exposed "early" at 26 months, by running at reduced V_{dd}.

Overall system The two simple mechanisms described above combine to provide a simple, yet effective circuit failure prediction technique advancing the state-of-the-art. Overall, Aged-SDMR works as follows and is depicted in Figure 1(e) and Figure 2. Program execution is broken into epochs of execution, with a *small window* at the start of each epoch executed in DMR mode. A light-weight mechanism is used to pair two cores arbitrarily and checking is done for every retired instruction. To implement virtual aging, every core is augmented with the capability to reduce its supply voltage by a small amount - it is the designer's choice, as it directly controls the timeliness of prediction. In practice, we simply leverage existing DVS mechanisms that can reduce V_{dd} by as little as 50mV. To implement *fault exposure*, at design-time (during the physical design phase), a delay-chain is inserted into each coarse-grained block. All fast paths are examined, and based on their delays, they are "connected" to different points in the delay-chain.

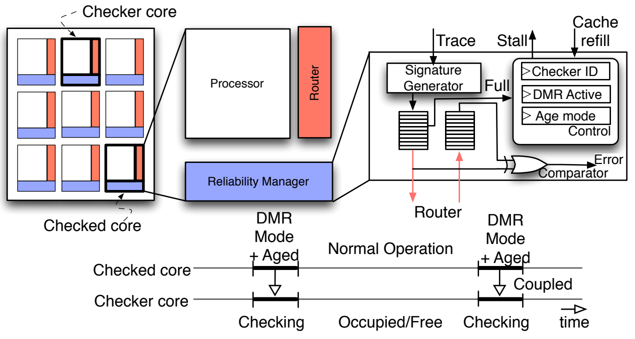


Figure 2: Design of Aged-SDMR

Paper contributions In this paper we describe in detail the design of Aged-SDMR. We also develop the general principles that a detection mechanism requires to leverage virtual aging and fault exposure to be repurposed as a prediction mechanism. We have implemented a full system prototype of Aged-SDMR on the OpenRISC processor running on an FPGA and evaluated with 10 SPEC benchmarks. We show that Aged-SDMR works in practice and effectively provides early warning of impending failure. It has area overheads of less than 9%, power overheads of 3%, and energy overheads less than 1%, and it can provide generality. This paper makes five contributions:

- Observation that detection and prediction can be unified by virtual aging to manifest wearout faults early.
- Circuit & microarchitecture mechanisms for fault exposure.
- Principles for repurposing detection techniques to prediction techniques.
- Full system design, implementation, and evaluation of fault prediction using Aged-SDMR that provides generality, low overheads, low design complexity and high accuracy.
- A methodological contribution with an end-to-end framework for timing fault studies on complete applications.

Implications Aged-SDMR was developed as a technique targeted at wearout. Its fundamental contribution and advancement to state-of-art is applicable more broadly. The key idea is to deterministically trigger DMR execution, while ensuring one of the duals operates in a stressed environment thereby guaranteeing errors occur first when DMR is active. We discuss these broader issues in the Conclusions outlining applications for reducing guardband, amplifying fault-mitigation techniques among others.

Paper organization The rest of this paper is organized as follows. Section 2 presents a primer on wearout. Sections 3 and 4 present the detailed design and implementation. Section 5 presents evaluation, Section 6 discusses related work and Section 7 concludes.

2 Primer on aging

This section gives an overview of three predominant wearout mechanisms and how they contribute to the increase in propagation delay through gates. For further details on wearout and modeling, we refer readers to [45].

2.1 Propagation delay in CMOS logic

The delay through a gate (t_d) is a function of the capacitance it drives (C), its size (L, W), the gate oxide capacitance (C_{ox}), the charge carrier mobility (μ_{eff}), the operating voltage (V_{dd}) and the threshold voltage (V_{th}) and is governed by the equation:

$$t_d = \frac{2LC}{W \mu_{eff} C_{ox} (V_{dd} - V_{th})^2} \qquad (1)$$

Most degradation mechanisms directly affect the V_{th}. This causes an increase in t_d, implying the gate becomes slower. Note that *reducing the* V_{dd} *has the same effect.* Increasing temperature reduces μ_{eff} and thus has a similar effect.

2.2 Wear-out mechanisms

Bias Temperature Instability (BTI) When a negative bias is applied to the gate at elevated temperature, holes/electrons migrate to the silicon-oxide interface. This causes an increase in threshold voltage V_{th} governed by, among other things, the switching activity of the device. Calimera et al. [11] report 10%-15% degradation in V_{th} over a year of circuit operation, which approximates to a delay degradation of 15% over 3 years of continuous operation.

Hot Carrier Injection (HCI) Occasionally, carriers (electrons or holes) gain sufficient kinectic energy and break into the the gate dielectric. Over time, this accumulation of carriers in the dielectric causes a degradation in V_{th}.

Time Dependent Dielectric Breakdown (TDDB) Long time application of a low electric field causes slow gate oxide degradation, which gradually increases the threshold voltage, causing a (*soft breakdown*). Eventually it leads to formation of a conductive path from the gate to the substrate, irreversibly breaking the gate (*hard breakdown*). Linder and Stathis characterize this in detail [32]. The notion of hard breakdown is traditionally thought of as the gate stopping to function, but is increasingly viewed also as having such high leakage that causes associated heating problems [24].

3 Design

In this section we present the design and implementation of Aged-SDMR. We first describe the system architecture and then details of the two underlying mechanisms for virtual aging and fault exposure. We then describe general principles for repurposing any detection technique, how to react to impending failures through the notion of an impending failure exception, and conclude with a discussion on practical implementation concerns.

Before getting into details, we review the meaning of circuit failure prediction. Microprocessor manufacturers use models and insert guardband to account for the expected degradation during chip's lifetime (say 8 years). Weibull distributions and bathtub curves capture failure distributions in the field [52]. Typically most chips (say 70% to 95% - exact values depend on maturity of process), will not have a single device that exhibits delay degradation exceeding the guardband of the chip during 8 years of operation. Circuit failure prediction pertains to the 5% to 30% of chips that exceed the delay guardband during their lifetime. The role of the predictor is to warn sufficiently in advance of an impending failure. The amount of guardband and percentage of non-failing chips are inter-related variables.

3.1 Aged-SDMR system organization

Figure 2 shows the system architecture of Aged-SDMR. The main additions to a Sampling-DMR system are i) voltage reduction of the checked core, which virtually ages only the checked code and ii) DMR error exception augmentation as an impending failure exception.

We summarize the system architecture that largely borrows from the work of Nomura et al. [37] and Wells et al. [61]. Each core in a multicore chip is augmented with a Reliability Manager (RM) that includes a FIFO and some small amount of state. It interacts with the processor accepting a trace of completed instructions converted into a signature and conditionally stalls it if the FIFO is full. A checked core drains its FIFO when possible by sending messages through an interconnection network to the checker core (saved in the Checker ID register). The checker core does similar operations to fill its local FIFO. In addition, it receives messages and writes them into the receiver FIFO. Whenever entries are available in the local FIFO and receiver FIFO, they are both popped and compared - differences indicate fault. To avoid memory incoherence and to avoid load-store ordering problems, the cores are synchronized on all cache refills (which includes dirty evictions). Since DMR is active for small parts of time (1% in our case), slowdowns do not impact performance.

We depict execution progress in Figure 2. To manage switching in and out of DMR mode and exposing a fixed number of available cores to the operating system, we assume the chip exposes a fixed number of virtual CPUs (VCPU) and has firmware to expose this to the OS [61]. The key benefit of the firmware VM layer is that it allows the VCPUs to be arbitrarily paused, allows quick transition in and out of DMR mode and makes available all physical cores to processes instead of reserving a dual for each core. Every VCPU is dynamically mapped to two physical cores (physical pair) by the firmware and the operating system assigns work to a VCPU. Each physical core is capable of operating in four modes, namely, checker mode, checked (or DMR) mode, non-DMR mode, and free mode. Execution of every process is in terms of epochs (100K to 5 million cycles, to define a system checkpoint to rollback to). Safetynet [50], Revive [42], ReviveIO [36] are all viable candidates for checkpoints. The firmware starts a process by marking one core to be in non-DMR mode and the other is marked available (free mode). When entering DMR mode, the firmware activates the checker core (by finding a core in free mode) and changes its mode to the checker mode. It then copies the entire architecture state (registers and cache-lines) from the checked core to checker core. It executes in DMR mode for the DMR period length. The checked core is always executed at a reduced voltage. If no error occurs, the checked core's mode is changed to non-DMR, voltage restored to nominal V_{dd} and the checker core is marked free. If an error occurs in DMR mode, the RM triggers an exception - the exception handler will restore the system checkpoint first. We specifically assume randomized sampling of the DMR windows. Alternatively the OS itself can be modified to handle these responsibilities.

In the following sections, we discuss the two underlying mechanisms of virtual aging and fault exposure.

3.2 Virtual Aging

Under normal degradation, the threshold voltage V_{th} of transistors increases, slowing down their operation. From equa-

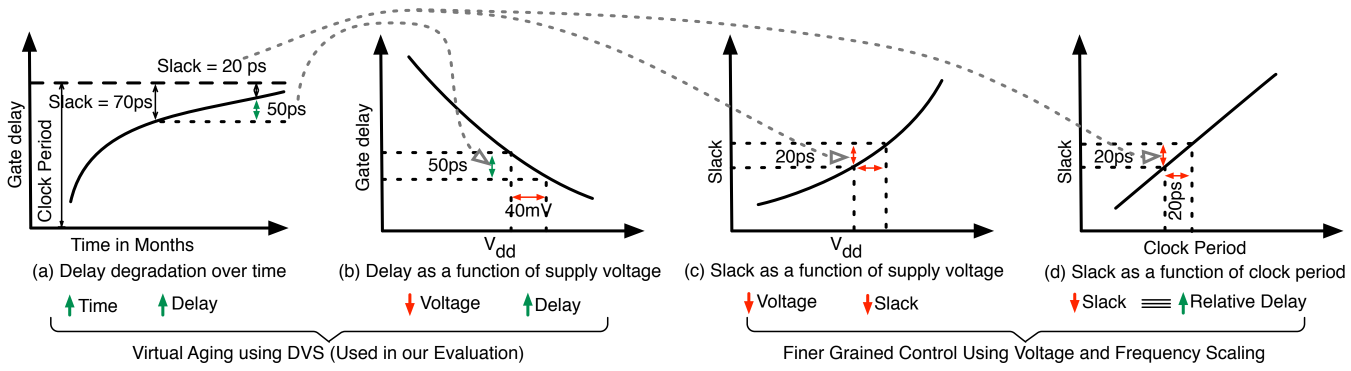


Figure 3: Delay increases as devices wearout; Reducing V_{dd} mimics aging. Slack may be removed by reducing V_{dd} or period

tion 1 we note that reducing supply voltage V_{dd} has the same effect. Figure 3(a)-(b) show this correspondence between delay from aging and delay from V_{dd} reduction. Restoring the operational supply voltage brings back the transistors to their previous state. We elaborate this with an example. Consider a device with nominal V_{dd} of 1.2V and V_{th} of 200mV at start of life. After 24 months of degradation say V_{th} is 250mV and after 36 months of degradation say V_{th} is 300mV. The concept of virtual aging is that, at 24 months, setting V_{dd} to 1.15V effectively creates a device that behaves like the 36 month old device ($V_{dd} - V_{th} = 0.9$). Reducing the supply voltage is a reversible and accurate method to force a processor to behave as if it has aged further. Equation 1 captures the impact of degradation (V_{th} increase) and V_{dd} on the delay of gates. We show this equation holds true emperically in Section 5.3.

Implementation Dynamic voltage scaling (DVS) is prevalent in commercial microprocessors and is controllable by operating system and firmware. We leverage this mechanism as-is in Aged-SDMR. When a core operates in checked mode, the DVS controller is set to deliver a lower supply voltage – 50 mV reduction is practical in modern designs.

Challenges Since virtual aging is achieved by reducing V_{dd}, this affects the minimum amount of delay that can be induced on gates i.e. the minimum amount of wearout that can be virtually induced. This introduces a relationship between the granularity at which V_{dd} may be controlled and how early a prediction may be made. In addition to changing the supply voltage, frequency (clock period) control may be used to improve the granularity. Figure 3(c)-(d) shows the direct correlations between gate-delay change to first path-slack and then from path-slack to clock period reductions. Through empirical measurements we can create a mapping from aging period in months to clock period reduction. Modern processors allow for clock period changes of as little as 25ps. This can be combined with voltage reduction to get fine grained control. For the technology node we consider, a 50mV reduction corresponds to 9 months of degradation and 80ps delay. Combining V_{dd} reduction with 50ps of clock period *increase* we can emulate, 30ps of aging delay which is 3 months of degradation. Dynamic fine-grained body biasing [1,54] which can be changed in $1\mu s$ to $10\mu s$ timescales can be used to effectively alter V_{th}, giving further control over granularity. In the evaluation, we restrict ourselves to voltage reductions alone since it is sufficient to demonstrate the utility of the concept.

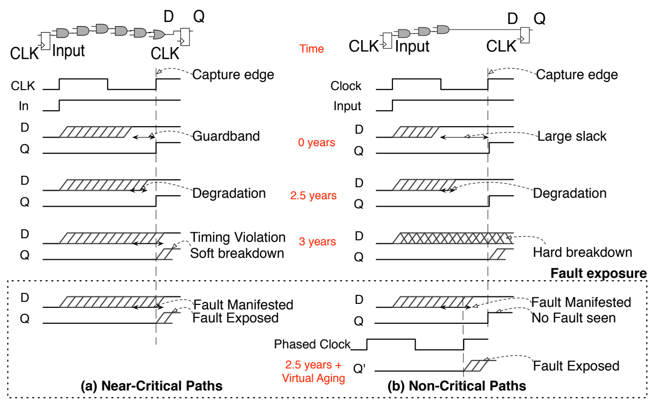


Figure 4: Signal Integrity in Near-Critical and Fast paths

3.3 Fault Exposure in Critical Paths

Terms *Fast-gate:* Gates that are exclusively part of paths whose delays are well below the clock period are called fast-gates. *Fast flip-flops and fast paths:* Flip-flops and paths where arrival times are well below the clock period. *Gate subcircuit:* A gate subcircuit is a set of gates in all paths that pass through a gate. We assume usual definitions of critical path and non-critical path.

Exposing faults in the critical path is straight-forward. Figure 4 (a) shows the effect of degradation in a critical-path assuming guardband is added to accommodate aging. As the chip ages, the delay increases and the guardband slack decreases. When the delay degradation overshoots the guardband (3 years in the figure), soft breakdown occurs. Under virtual aging, the additional delay in gates that fall in near-critical paths show up as faults at the flip-flops they drive. This causes a bit-flip (or metastability) at the output of the flip-flops that may propagate to cause an architectural state corruption. These faults are exposed, *with no modifications required to the processor.* Figure 5 shows an example circuit block highlighting the fact that the critical path is left unmodified.

3.4 Fault exposure in Non-Critical Path

Non-critical paths introduce subtle challenges because gates that are *exclusively* on non-critical paths (fast gates) may directly degrade to hard breakdown without ever manifesting as a delay fault. On the OpenRISC and OpenSPARC processor, respectively, 39% and 30% of all logic gates are such fast gates. *Note that these gates are as likely to wearout*

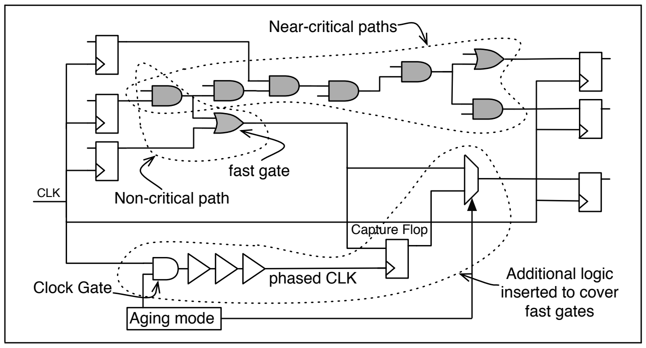


Figure 5: Circuit and changes required to capture all faults

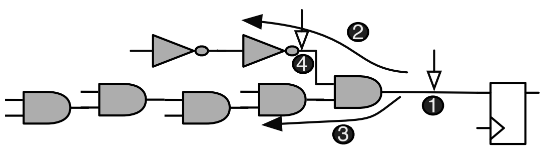


Figure 6: Where do capture flip-flops go?

as the gates on critical paths. To the best of our knowledge, all prior prediction techniques leave these gates uncovered and they will unknowingly transition to hard breakdown. To expose these faults, we develop a novel mechanism.

Concept In non-critical paths, the delay through gates is much smaller than the clock period. This excess slack masks normal and virtual aging induced degradation. Figure 4 (b) illustrates this effect. At the time of manufacture, there is a large inherent slack. After a period of use, significant delay degradation occurs. However, there is sufficient slack and the flip flop still captures the correct value. Eventually, the gate goes into a hard breakdown. At this time, the output of the circuit is indeterminate. Before going into hard breakdown and with virtual aging enabled, a delay degradation goes unnoticed as it is masked by the excessive slack. By capturing the signal at an appropriately "phased clock", the fault can be exposed.

Implementation We implement the phased clock idea at a relatively coarse granularity of pipeline stages. In each stage, we begin by creating a number of phased clocks by passing the clock through a delay chain. Using timing information gathered from the synthesis phase, we determine the nominal delay through the non-critical paths. The phase of the clock that best captures the degradation (see Figure 4) for each path is then chosen. A "capture flip-flop" triggered by this clock is introduced at the end point of each path. A mux is inserted to override the signal with a captured fault if a core is in checked mode. This facilitates power/clock gating the delay chains and capture flip-flops when the core is in other modes. Figure 5 shows the additional circuitry added to cover one such fast gate. The additional mux in the circuit path causes a small increase in delay. This does not pose a concern as these paths are, by design, non critical. This also enables the use of high threshold, low power gates for the additional circuitry. Note that in non-checked mode, the capture flip-flop is bypassed. Since, the capture flip-flop is out of phase and its edge is far earlier than the regular clock, in checked mode, we are not introducing an additional clock cycle even though the signal passes through an additional flip-flop.

Figure 6 illustrates the need for inserting the capture flip-flop at the gate level granularity Consider two non critical-paths which have the same end point ❶. Inserting a capture flip-flop here will capture the degradation in path ❸. However, the fast path ❷ is "faster" than ❸. We insert a capture flip-flop at ❹ specifically for gates in path ❷.

Tradeoffs/Challenges The phase of the clock can be incremented at the granularity allowable by the delay elements. Thus faults at a few gates whose delays are closer to the quantized delay are captured at lesser degradation than others. Our end goal is a prediction window measured in hours and days, hence this impact is overall insignificant. In our design, we implement one delay chain per pipeline stage. We derive *multiple* phased clocks using a *single* delay chain per module. At design time, after timing closure, we determine the worst case delay through fast paths. We choose the clock phase that best captures degradation, for example, the worst case delay + 14%. This is done in design time on paths with large slacks – the design complexity is low in the sense changes are only to non-critical paths. It can be done at various coarser or finer granularities and can even be folded into clock-tree generation.

Another issue for the non-critical paths is slack distribution. Even in highly optimized production designs, fast paths do exist. Bowman et al. [9] show the distribution in an Intel microprocessor. While Vdd reduction would cover most (near critical) paths, additional capture logic is essential to cover the fast paths – most prior work focuses only on near-critical paths.

3.5 Principles required in the detection mechanism

Virtual aging coupled with fault exposure can be combined with any detection mechanism. We now develop the principles a detection system must provide for it to be compatible.

Principle 1: A checker. Introducing faults early is risky. The detection mechanism must have a *robust checker* - ideally providing 100% fault coverage.

Principle 2: Transition in and out of aged mode. The aging mode is activated only for short periods of time. The detection mechanism must facilitate being *turned on and off* without large overheads.

Principle 3: Recovery. When a virtually manifested fault is exposed and then detected, a mechanism to *recover* must exist. The knowledge and capability that we precisely control when we enable virtual aging can be exploited.

We find Sampling+DMR to best fit these requirements. Table 1 shows how four other state-of-the-art detection techniques can be repurposed for prediction by manifesting faults using virtual aging and exposing them using our mechanism. In general they are enhanced with this hybridization compared to their original detection-only approach: Argus and DIVA need to be turned on only during sampling windows of prediction, reducing their power overheads. SWAT can be augmented with shorter epochs since the performance overheads get gated by sampling. Full DMR's power overheads also get gated by sampling. We note that some of these techniques were designed to capture transient faults also, which sampling curtails.

	Pipeline Stage	Functionality	Slack (ns)	Notes
G1	PC Generate	Next PC value (bit 3)	0.0003	Part of mux that chooses next PC based on decoded branch opcode.
G2	ALU	Adder (bit 31)	0.051	MSB of addition (ripple carry adder). Switches when input operands change, even if not executing an ADD instruction.
G3	Instr Fetch	Instr from ICache (bit 30)	0.026	Gate at output of instruction cache, critical as cache read cycle delay is high.
G4	LSU	Write back on load (bit 0)	0.520	LSU data forward from data cache. Value latched before forwarded: non-critical.
G5	Decode	Extract Immediate Value (bit 0)	0.943	Extract immediate value from instruction on decode Path contains only buffers: non-critical.

Table 2: Gates analyzed in our evaluation

Detection Technique	Principle 1	Principle 2	Principle 3
Argus	Yes	DVS + Sampling	Pipeline flush
DIVA	Yes	DVS + Sampling	Pipeline flush
SWAT	Symptoms	DVS + Sampling	Checkpoint
Full DMR	Use checker	DVS + Sampling	Pipeline recovery

Table 1: Re-purposing other detection techniques

3.6 Impending failure exception

We propose a new “Impending failure exception” to convey a failure prediction to the operating system. The operating system may handle this exception in a lazy manner in many possible ways: decommission the core, reserve the core only for non-critical application code [30,44], trigger wearout mitigation techniques [12, 27, 54, 55], or salvage the core [41].

3.7 Other practical concerns

Can errors escape? During sampling windows, virtual aging manifests faults in conjunction with DMR to detect the errors. By doing so, we guarantee that when the errors first occur, we detect them. As we have full control over when the faults are manifested and have full coverage DMR mechanism in place to detect errors, Aged-SDMR does not suffer from missed errors escaping - an important distinction to Sampling-DMR. An issue arrises if the underlying fault does not first present as a delay degradation. We discuss this as Question 7 in Section 5.

Could Aged-SDMR make wrong predictions? We can look at two possibilities. First, consider false positives — Aged-SDMR predicts failure, but no “failures” occur. A failure is predicted only when a wearout fault is impending in a gate and application(s) activates the gate during sampling. Once a failure is predicted, a different application(s) that does not use this gate will not encounter errors. In this regard, Aged-SDMR is at least as good as prior art. Second, false negatives – Aged-SDMR fails to make a prediction before breakdown. This is indeed a fundamental issue that we analyze in detail in our results. We show that this is exceedingly rare and can be avoided by common design practices. In the above cases, the change in the application behavior has to be radical. For example, for a false negative, an application(s) that does not use the floating point unit does not detect wearout in that unit. After breakdown, an application that uses the floating point unit will encounter unpredicted errors.

Relation and improvements over BIST coverage Existing BIST techniques are insufficient for the following reasons. (i) Using limited test vectors, BIST/DFT can detect stuck-at faults efficiently. Nomura et al. [37], explain that for BIST like techniques to achieve 99.5% delay fault coverage, a large test vector is required and the processor must be taken offline for 10s of seconds. As any gate in a processor may breakdown, 100% coverage is necessary which BIST cannot provide. (ii) Online environment issues, like application behavior triggering voltage noise are not captured by offline techniques. (iii) Delay degradations in fast paths (prior to hard breakdown) will never cause timing failures in BIST test vectors, since they have large slack - additional fault exposure logic is necessary. However, adding fault-exposure to BIST is insufficient, because covering all gates/paths means significantly longer vectors.

Execution perturbation We define execution perturbation as the difference to program execution (time and ordering of events) compared to a hypothetical run/system without prediction techniques. On epochs where no impending failure exception is raised, no perturbation is introduced beyond what the DMR execution introduces on sampled windows. On epochs where it is raised, restoring to a checkpoint introduces “avoidable” perturbation, because the original processor has not yet failed.

Interaction with process variation and voltage noise An issue that is universal to all wearout prediction techniques is interaction with voltage noise and process variation related guardband. The process related addition in guard band is static. Regardless of whether speed binning [14], techniques like body biasing [57], or variation-tolerant power optimizations [15, 29] are used for a given chip, its influence remains unchanged during life time and hence does not impact Aged-SDMR. The guardband to accommodate operating condition variations, in particular, voltage noise and temperature (which affects carrier mobility and hence gate delays) is dynamic. Hence, it may lead to changes in the timing slack at random clock cycles even when all other factors are constant. This dynamic variation may hinder exposing the fault under virtual aging. When operating a core under checked mode (with virtual aging), a variation – for example, a positive surge in supply voltage – may cause the degradation to be masked. This poses a challenge in capturing the fault. However, such noise is infrequent and/or intermittent, and hence in worst case, the end effect is additional time to predict failure.

Relationship to transient faults Our philosophy and claim is that a missed permanent fault ($\leq$ 100% coverage) results in unbounded missed errors in the field because of recurring errors from the fault, while $\leq$ 100% transient fault coverage is more benign. We thus argue that it is best to decouple treatment of permanent and transient faults and design for 100% permanent fault coverage. For example, Shoestring [17] can be used if 100% transient fault coverage is not necessary. Alternatively, physically-based targetted transient fault detection provides 100% coverage [59].

Number of fast gates The additional logic inserted to cover fast gates contributes to an increase in logic area. We argue that this is not a significant design concern in future designs because (i) tuning for power reduction replaces gates in non critical-paths with their low power (slower) counter parts – reducing the number of fast gates. (ii) The clock phase shifting logic may be implemented with relaxed skew requirements – this adds minor imprecision to the prediction horizon. (iii) The additional logic is added to paths with large slack – obviating need for further timing optimizations.

4 Implementation

In this section we describe our prototype implementation of an Aged-SDMR system. Our goal in building this prototype is to understand its effectiveness and overheads of the technique. Due to the difficulties in inducing actual wearout, we develop a delay-aware simulation to get fault vectors which are then injected into the prototype system.

4.1 Baseline Processor

We use the the OpenRISC OR1200 [38] - an ASIC proven microprocessor with full system stack support. Using a 32nm technology library, the processor was synthesized to fit timing with a clock period of 1.2 ns and a guardband of 0.08 ns. To measure fault rates and understand wearout effects, we pick five representative circuits for analysis. We run these through detailed SPICE simulations to understand delay degradation. This strategy is similar to picking fault sites in functional simulator based studies. We considered diversity across pipeline, a mix of critical and non-critical paths, and high switching activity gates (most likely to fail). Table 2 lists the gates with characteristics. Gates 4, 5 are *fast gates.*

4.2 Aged-SDMR Prototype system

Virtual Aging To model virtual aging, we use the gate library design specification and run delay aware simulations and SPICE simulations with the correspondingly reduced V_{dd} — we determined that 50mV corresponds to 9 months.

Fault Exposure We wrote custom scripts to automate adding the circuitry required to capture faults in non critical-paths – the delay chain, the capture flip-flop and the mux as shown in Figure 5 – to the netlist. This gives us a version of OpenRISC with fault exposure mechanisms added to it. We then used delay aware simulation to determine when a flip-flop (either critical or non-critical) captures an incorrect value due to delays. Running logic simulation or delay-aware simulation for a full processor on full benchmarks is impractical, and an FPGA cannot perform delay aware simulation, nor is overclocking or under-volting an FPGA representative of wearout or aging an ASIC processor. Hence we developed a split strategy - *note the below steps are for evaluation only and are not required in a production system.*

For each of the five gates described in Table 2, we extracted a "gate subcircuit" – that includes all gates that are in the gate's fan-in and fan-out, including the clock phase shifting and capture logic. While running benchmarks on an unmodified OpenRISC processor mapped on an FPGA, we saved the input sequence driven to the gate subcircuit during sampling windows. We then fed this as inputs to the gate subcircuit's delay aware simulation which produced a fault vector, marking the cycles in which timing faults occurred.

Technology	
• Digital Cell Libraries	Synopsys 32nm multi-V_{th} library
• Transistor models	Synopsys 32nm PDK (level 54)
Tools	
• Degradation Modeling	HSPICE + MOSRA
• Delay-aware Sim	Synopsys VCS
• Area/Power Analysis	Design Compiler
• FPGA Prototype	Xilinx Zynq 7020
Benchmarks	
• SPEC CINT2000	gzip, vpr, mcf, parser, vortex, bzip†
• SPEC CFP2000	art, equake, ammp, mesa†

† Compiled using uClibC for OpenRISC. Other benchmarks were incompatible. File-IO modified as memory ops to run on bare-metal.

Table 3: Infrastructure used in our evaluation.

Aged-SDMR prototype We implemented an emulation of a full Aged-SDMR system by time-slicing an FPGA to run two copies of OpenRISC 1,000 cycles at each step. We sampled for 100,000 cycles in epochs of 10 million cycles (1% DMR). We also implemented a simple reliability module. Outputs, representing the trace of retired instructions is buffered in an internal FPGA RAM. Virtual aging on the checked core is modeled by injecting the fault vector from the previous experiment. For prototyping we use the Xilinx Zync 7020 FPGA which includes an ARM cortex A9 core, which we utilize for orchestrating the time-slicing.

We do *not* model the interconnection network mapping of multiple cores, firmware layer or VCPUs. We instead time-slice two cores on the same FPGA. The implication is that our prototype does *not* model performance degradations — but as described previously, Sampling-DMR's overheads are 1-2% with 1% sampling.

5 Evaluation

We describe our methodology, and then present results organized as answers to a set of evaluation questions.

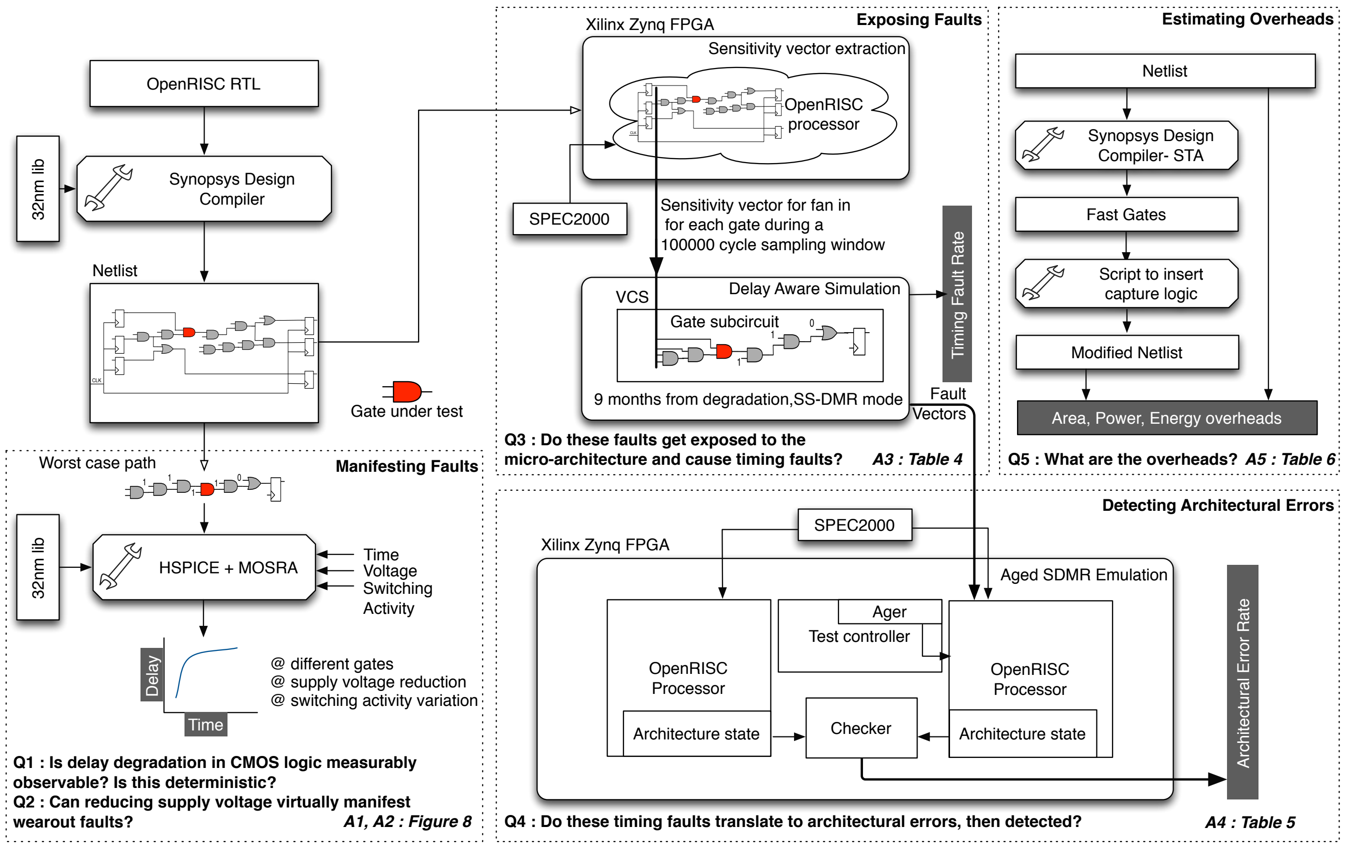
5.1 Goals

Our goal of understanding wearout and Aged-SMDR's effectiveness is organized around the following questions.
Q1. Is delay degradation measurably observable?
Q2. Can voltage reduction virtually manifest wearout faults?
Q3. Do the manifested faults get exposed to the μarch and cause timing faults?
Q4. Do the faults exposed to the microarchitecture translate to architectural errors, then detected?
Q5. What are the overheads?
Q6. What is the delay to predict?
Q7. When does this technique provably fail to predict wearout?

5.2 Methodology

Figure 7 shows the flow of data across our evaluation prototype, tools used to obtain the data and questions they help to answer. The general philosophy is as follows: (1) Use SPICE + MOSRA to evaluate any gate level effects. These tools when used with the advanced MOS models provide accurate measurements. (2) Use gate level delay aware simulations to check for timing faults. (3) Where actual run time data is required, use full system emulation running on the FPGA setup. Table 3 lists the technology, tools and benchmarks we use in our evaluation. Note that we refer to five representative gates we track as gates G1-G5.

Understanding degradation (Q1) We modeled the gate subcircuits for G1-G5 in SPICE and obtained delay degra-



OpenRISC RTL
32nm lib
Synopsys Design Compiler
Netlist
Gate under test
Worst case path
Manifesting Faults
32nm lib
HSPICE + MOSRA
Time
Voltage
Switching Activity
Delay
Time
@ different gates
@ supply voltage reduction
@ switching activity variation
Q1 : Is delay degradation in CMOS logic measurably observable? Is this deterministic?
Q2 : Can reducing supply voltage virtually manifest wearout faults?
A1, A2 : Figure 8
Xilinx Zynq FPGA
Exposing Faults
Sensitivity vector extraction
OpenRISC processor
SPEC2000
Sensitivity vector for fan in for each gate during a 100000 cycle sampling window
VCS
Delay Aware Simulation
Gate subcircuit
9 months from degradation,SS-DMR mode
Timing Fault Rate
Fault Vectors
Q3 : Do these faults get exposed to the micro-architecture and cause timing faults?
A3 : Table 4
Estimating Overheads
Netlist
Synopsys Design Compiler- STA
Fast Gates
Script to insert capture logic
Modified Netlist
Area, Power, Energy overheads
Q5 : What are the overheads?
A5 : Table 6
Detecting Architectural Errors
SPEC2000
Xilinx Zynq FPGA
Aged SDMR Emulation
Ager
Test controller
OpenRISC Processor
Architecture state
Checker
OpenRISC Processor
Architecture state
Architectural Error Rate
Q4 : Do these timing faults translate to architectural errors, then detected?
A4 : Table 5

Figure 7: Setup of all the experiments and inter-relationships between them

dation with age using MOSRA and occurrence of soft and hard breakdown. We repeated the experiment by varying the switching activity to bring out its influence on degradation. *worst case paths through 5 gates × 60 months × 5 activity factors = 1500 runs.*

Manifesting Faults (Q2) We ran the SPICE+MOSRA setup above with many reduced supply voltage settings to determine how well voltage reduction mirrors aging. We report results for 2 V_{dd} settings. *worst case paths through 5 gates × 60 months × 2* V_{dd} *= 600 runs.*

Exposing Faults (Q3) Using the fault exposure setup from our prototype implementation (Section 4.2), we measured the fault rate for each benchmark. To measure the variability across execution phases of a benchmark, we ran this experiment in three different 100,000 cycle sampling windows. *5 gates × 3 sampling windows × 10 benchmarks = 150 runs.*

Detecting Faults as Architectural Errors (Q4) We inject faults based on fault vector from previous experiment into Aged-SDMR prototype, detect errors and measure error rate. In total, across three windows while executing each of the ten benchmarks on the full processor, this experiment covered 21 to 9862 faults injected per gate. *5 gates × 3 sampling windows × (21–9862) faults (across 10 benchmarks) = 399453 runs.*

Estimating the Overheads(Q5) and Q6/Q7 We used synthesis tools to estimate overheads on OpenRISC and the OpenSPARC processors. For Q5 alone we consider a second processor OpenSPARC to understand sensitivity to design optimizations. We formulated thought experiments and mathematical analysis to answer **Q6** and **Q7**.

5.3 Results

Our results are presented as answers to the seven questions.

Understanding Degradation (Q1) Figure 8(a),(b) shows degradation over time (simulated for 60 individual months of continuous operation at 30° celcius with full switching activity) of the worst case paths through the five representative gates. As expected, delay grows monotonically[2]. When the delay exceeds clock period (α in graph), soft breakdown manifests as timing faults. Gate G4 transitions to hard breakdown at 56 months. To capture influence of switching activity, we show details for one gate G1 in Figure 8(c), with switching activity rates varied between 10%-100%. It is clear switching activity plays a significant role in wearout demonstrating the need for in-situ techniques.
Answer 1: *Delay degradation in CMOS logic is measurably observable, and is dependent on factors including the switching activity and hence cannot be statically determined.*

Manifesting Faults (Q2) To demonstrate virtual aging we show G1 in detail. Figure 8(d) shows delay again over a 60 month simulation with V_{dd} and reduced $V_{dd} = 1.15V$. We see that we can force the onset of the soft breakdown early.
Answer 2: We see that *reducing* V_{dd} *mimics aging.* In this case a 50 mV (4.1%) reduction corresponds predicting up to

[2]Each point plotted is delay of circuit at one instance in that month - the variations are due to MOSRA's internal modeling of noise.

	Gate 1				Gate 2				Gate 3				Gate 4				Gate 5			
	AF%	**Faults**			**AF%**	**Faults**			**AF%**	**Faults**			**AF%**	**Faults**			**AF%**	**Faults**		
ammp	33.3	3324	3323	3222	46.2	6653	4618	4613	66.6	6660	6660	6661	33.3	3330	3332	3330	65.2	6545	6521	6522
art	66.5	0	47	0	33.2	0	6653	6654	21	24	0	0	66.4	6641	6617	6639	1.2	0	42	0
bzip2	11.9	3442	2610	7474	3.6	1488	2901	657	10.0	1355	308	1355	4.2	408	682	155	22.5	220	85	397
gzip	15.8	6472	1194	2154	2.9	190	476	221	4.4	541	245	541	1.8	190	239	110	9.3	87	63	128
mcf	16.6	8018	0	5382	32.8	5062	532	4257	13.3	531	2933	530	10.2	1054	531	1468	2.3	320	364	0
mesa	13.1	9862	1020	3466	17.6	155	4160	953	20.9	2816	637	2815	6.7	120	1558	352	26.8	488	1397	6182
parser	23.3	5518	6588	8774	26.2	2133	3886	1875	20.7	2493	1226	2494	12.2	929	1818	901	43.8	3799	2572	6782
quake	16.2	4720	2421	2264	14.2	833	1166	2270	13.6	1102	1858	1112	9.3	1370	487	934	39.7	6410	4309	1208
twolf	66.4	6648	6648	6635	22.1	0	0	6653	1.7	0	0	68	60.6	4988	6654	6654	60.9	4989	6653	6655
vpr	16.7	2530	5354	8756	14.7	2962	1054	396	24.7	3054	3058	1296	12.9	1502	1668	716	47.4	5789	2163	6274

Table 4: Timing faults in three 100000 cycle sampling windows seen when running at reduced voltage 9 months before failure. **AF** indicates the rate at which the gate toggles

Benchmark	**G1**	**G2**	**G3**	**G4**	**G5**
ammp	1.6%	3.1%	5.1%	1.4%	1.4%
art	0.02%	2.7%	0.01%	2.6%	0.01%
bzip	2.3%	1.2%	0.9%	0.2%	0.07%
gzip	1.5%	0.03%	0.4%	0.04%	0.01%
mcf	3.4%	3.1%	0.9%	0.7%	0.02%
mesa	2.2%	1.0%	1.2%	0.09%	0.8%
parser	4.3%	1.3%	1.9%	0.5%	1.5%
quake	1.9%	0.9%	0.8%	0.2%	1.3%
twolf	3.3%	1.1%	0.02%	4.3%	1.9%
vpr	2.6%	0.8%	2.1%	0.7%	1.6%

Table 5: Error rate - detected by Aged-SDMR

nine months in advance.

Exposing Faults (Q3) Table 4 lists the toggle rate (switching activity) and faults that virtual aging generated. The three fault columns correspond to three 100000 cycle tests run at different phases of the benchmark application. This test used the processor running benchmarks on the FPGA to extract input sequences, that are then fed to delay-aware simulation of Aged-SDMR netlist to get timing faults — for G1-G3 in in-circuit flip-flop and G4 and G5 in inserted capture flip-flop.

Virtual aging induces timing faults at the rate of between 0 – 9.8%. While we capture no faults in some sampling windows, faults appear eventually. An interesting case that showcases this is running the art benchmark while looking for faults to occur on gate G1. During two of the sampling windows, the phase of the program was such that the input sequences that activated the worst case path through G1 did not appear. In a third window, we observed timing faults being captured.

Answer 3: For an impending soft or hard breakdown, *when running at Aged-SDMR mode with workloads, timing faults appear indicating an impending failure.*

Detecting Faults as Architectural Errors (Q4) Table 5 shows architecture error rate across sampling windows on our Aged-SDMR prototype. Using the fault vector from the previous experiment, bit flips were injected in the gates. A check between a reference trace – of PC, register writes – and the trace with a fault injected was used to detect an architectural fault. Note that this experiment is determining the independent error-rate – for each fault, we have one fault-injected run. For example, for G1 on ammp, we run 3324 experiments. Each run represents one epoch of execution.

Answer 4: *Faults introduced in Aged-SDMR mode translate to architectural errors and can be caught without escapes.*

Estimating the Overheads (Q5) Table 6 quantifies the area, power, and energy overheads of adding fault exposure to OpenRISC and OpenSPARC. We show overheads for "logic only" (excluding SRAM structures, caches and TLBs, and register files) and for the full core[3]. The difference in area and power overheads across cores is due to the number and percentage of fast gates. From the cumulative distribution of worst case delay through gates we observed that 39% and 30% of gates are *fast gates* in OpenRISC and OpenSPARC. Note that by design, Aged-SDMR's overhead logic is active only 1% of the time. Power and energy overheads (including energy consumed by the checker core) are $\leq$ 1% to 3%.

	OpenRISC		OpenSPARC	
	Logic	Core*	Logic	Core†
Gates on fast paths	39%		30%	
Area Overhead	28.95%	8.9%	22.18%	6.8%
Peak Power Increase	3.2%	2.54%	2.21%	0.99%
Energy Increase	0.9%	0.7%	1.02 %	0.21 %

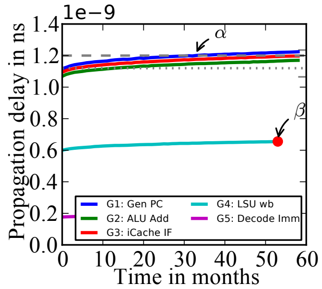
$*$ OpenRISC: with 8KB L1 Instruction & Data cache. 64 entry TLBs.
† OpenSPARC: with 16K L1 Instruction & 8K L1 Data cache.

Table 6: Aged-SDMR overheads for fault exposure

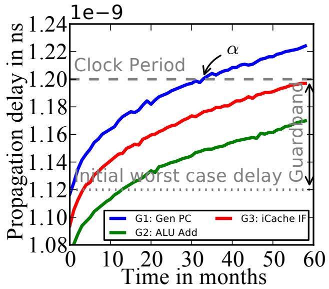
Answer 5: *Aged-SDMR has small area (8.9%), power (2.54%), and energy overheads (0.7%).*

Delay to predict (Q6) We can guarantee effectiveness and Aged-SDMR's latency to predict in mathematical terms and defect rate. We define a defective chip as one in which the prediction mechanism fails to raise impending failure exception before wearout occurs and results in *missed errors.* For a defect rate of D_R, and random sampling rate S, the upper bound on the number of *epochs* that need to be observed is $\frac{log_e D_R}{S}$; which is = 2072 epochs for $D_R = 10^{-9}$ (an extremely low defect rate) and 1% sampling. This is a result that is independent of fault occurrence patterns and architecture error patterns as proven by Nomura et al. [37]. Virtual aging and random sampling guarantees that faults *first* appear in DMR execution in any epoch and thus are not missed (do not escape). For a given clock frequency, epoch size, and per epoch error rate, the worst case # days to predict is $\frac{2072 * \frac{\text{epoch size}}{\text{frequency}} / \text{per epoch error rate}}{60*60*24}$. Assuming a 5 million cycle epoch, and 3 GHz clock frequency, and 10^{-4} per epoch error rate (lowest we observed - Table 5), this amounts to 0.4 days of activity to predict [4].

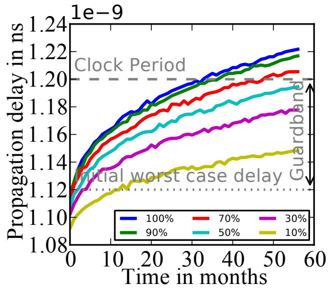
[3]OpenRISC especially has somewhat inefficient logic implementation - its SRAM and register files contribute only 22% to core power. We believe in a typical commercial-grade processor, overheads of Aged-SDMR will be even less since storage structure contribution is more.
[4]In contrast, Nomura et al. report a worst case latency of 78 seconds to sustain $D_R = 10^{-9}$. The discrepancy is that their latency is for the worst particular error occurrence pattern observed in their benchmarks (determined with a a 3-state markov model). Our result holds for any possible pattern since we are counting epochs.



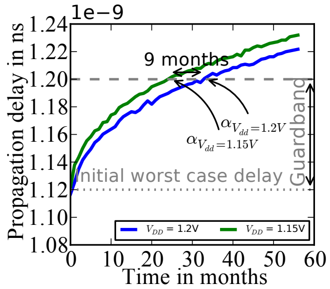
(a) Delay degradation over 60 mo



(b) Zoomed version of (a)



(c) Switching activity→degradation



(d) Virtual aging on G1 at 1.15V

Figure 8: Delay degradation and virtual aging

Technique	**Overheads**		**Time to**	**Prediction**
	Area	**Power**	**predict**	**horizon**
Online Wearout Prediction [6]	4.6%†	8.6%†	4 days	2yr - 4 days
Wearmon [62]	∼14%‡	NR	Varies	NR
FIRST [49]	NR	0%	1 day	9mo-1 day*
Aged-SDMR	8.94%	3.2%	0.4 days	9mo-0.4 days

† : For every 8 signals monitored; ‡ : Rough estimates from FPGA utilization numbers reported by the authors; NR : Not Reported; ∗ : Assuming a virtual aging mechanism similar to this work.

Table 7: Aged-SDMR compared to state-of-the-art

Answer 6: This result can be interpreted in two ways. First the longest latency to predict is 0.4 i.e. after the first onset of a fault under virtual aging, the impending failure exception will be triggered in all but 1 of 1 billion chips within 0.4 days. Alternatively, as long as voltage reduction (or with clock period reduction) causes a virtual aging of greater than 0.4 days, Aged-SDMR will predict the fault without faults escaping.

When does this technique not work (Q7) Faults that do not start as delay faults cannot be predicted by Aged-SDMR (electromigration and package level faults like thermal cycling). Given device faults that do adhere to this behavior, for Aged-SDMR to fail, there must be fault sites that have high switching activity and hence wearout during a program's (or set of programs) execution but *not* produce any effect on the architectural trace - for example integer benchmarks may do this to the floating point pipeline. For the fault to then get exposed, the fault site is exercised at some later point after wearout has occurred. If the fault-triggering workload runs after delay has exceeded guardband, Aged-SDMR will fail. If > 0.4 *days* of life remains, Aged-SDMR will still predict correctly. Regardless the masking scenario is exceedingly rare in commercial designs because power/value gating is common design practice to avoid unnecessary switching.

Answer 7: In only very rare scenarios is Aged-SDMR provably ineffective - these senarios can be avoided with common design practices.

Comparison to other prediction techniques Table 7 compares Aged-SDMR to three state-of-the-art techniques [6,49, 62]. Recall the previous techniques do not provide generality and leave 30% to 40% of gates uncovered. Aged-SDMR is comparable if not better on other metrics and in addition provides generality.

Limitations We conclude with a discussion of three evaluation limitations; design limitations were addressed in Section 3.7. Our prototype implementation and evaluation did not consider performance overheads, since the full multicore system was not implemented because it is not our work's focus. DMR period is active for 1% of time, and previous results on performance impact of Sampling-DMR show it is 2% and hence not necessary to re-demonstrate. Our experimental evaluation did not trigger worst case error patterns on a very low error rate and a burst of errors in the 2072nd epoch - so we did not empirically demonstrate the 0.4 days limits. Triggering controllable error patterns is hard and instead we perform sound mathematical analysis. We avoided sensitivity studies with voltage reduction + clock period reduction, since it requires careful and measured calibration of V_{dd} to gate delay. Instead we rigorously evaluated using 50mV which sufficiently demonstrated Aged-SDMR's value.

6 Related Work

We discussed much related work in the introduction. We summarize others here. Modeling work [3,51,58] and mitigation techniques [7,21,55] are orthogonally related. Of the canary based [56,60] and in-situ [3,9,16,39,49,62] prediction techniques, FIRST [49] and WearMon [62] are most closely related and we revisit them here focusing on similarities to our work. FIRST introduces the idea of "stressing" the processor in its test-vector injected mode without describing exactly how, while WearMon eliminates guard band. Neither formalizes the idea of virtual aging and the interaction to fast gates and both suffer from generality since they selectively go after critical paths to curtail overheads.

7 Conclusion

This paper proposed Aged-SDMR which unifies fault prediction and detection to create a technique that provides low overhead, generality, high accuracy and low design complexity. It includes the concept of virtually aging the processor and then uses Sampling-DMR to detect faults exposed. To guarantee generality and coverage of all faults/gates, Aged-SDMR uses a novel mechanism for fault exposure. This paper showed the design principles and reported on a full prototype implementation. We demonstrated Aged-SDMR works in practice using an FPGA demonstration on 10 full SPEC benchmarks and has 8.9% area overhead and less than 1% energy overhead. We conclude the paper with thoughts on the fundamental capability of Aged-SDMR, its implication within processor reliability and broader implications.

Fundamental capability By providing low overhead, generality, high accuracy and low design complexity, Aged-SDMR could serve as an important component for future fault dominated technologies. Aged-SDMR is a simple idea making it attractive for practical deployment but provides unprecedented capability for meeting all four design constraints.

Implication for reliability In addition to providing a solution for reliability, an important contribution of our work is the identification of the fundamental issue that gates in non critical-paths are as likely to fail as gates in critical paths - and have to be covered. Almost every related work in literature implicitly or explicitly ignores these gates – which may amount to 30% to 35% of all gates in a microprocessor. Aged-SDMR proposes a low overhead solution to cover these gates.

We developed Aged-SDMR as a prediction technique to deal with wearout related end-of-life failures. Stepping back from this specific use case, the virtual-aging combined with Sampling-DMR provides the general capability that faults can be predicted/detected before errors leak into the system. A disruptive implication is that a machine equipped with Aged-SDMR can dynamically manage guardband added for handling uncertainty. For example, voltage noise and temperature are sources of uncertainty. If the checker-core can be located physically far apart, it could be impervious to noise and temperature effects as the checked-core. Furthermore, time scales at which noise and temperature change are likely to be much larger than the epoch sizes of DMR thus allowing Aged-SDMR to behave as a detection mechanism. Another example is to deploy Aged-SDMR as a way to aggressively manage guardband for static variations. Conventionally, the operation point of a processor is determined after fabrication through VLSI testing and some guardband is added. Alternatively, Aged-SDMR can be used in the field, to find the V_{DD}/clock period such that we get fault free execution for the next 6 months only (starting from a fresh chip). This voltage will vary between different chips depending on the process variation. In effect, we could use Aged-SDMR to operate a microprocessor at exactly the point where we know no failures will occur in the next few months.

More generally, looking at the broad scope of prediction, detection and mitigation, Aged-SDMR serves as an effective prediction and detection technique. Thus it's presence can amplify the role of mitigation techniques and allow mitigation techniques to be very aggressive, since they can operate under the assumption that there is a fail-safe prediction/detection technique which guarantees no incorrect operation will occur. For example, mitigation techniques like Facelift [55], Bubblewrap [25] and others [48] can be configured to be aggressive in terms of energy efficiency.

Broader implications Beyond reliability, the core idea of Aged-SDMR is that it allows controlled triggering of DMR and guaranteeing that errors occur first when in DMR mode, thus providing extremely lightweight detection capability. In our work, we utilize this for predicting wearout. More generally, our work enables a new paradigm to deal with other sources of uncertainty – by allowing predictors for various other phenomena to control periods during which DMR is active. Aged-SDMR ensures error free execution as long as the prediction technique guarantees no false negatives. This opens up opportunities to rethink many techniques for uncertainty beyond just reliability including perhaps software reliability due to bugs etc.,

Overall, Aged-SDMR provides a new perspective on the decades' old idea of dual modular redundancy and could trigger rethinking the many current uses of DMR and also creating new uses.

8 Acknowledgments

We thank the anonymous reviewers, the Vertical group, Kewal Saluja, Marc de Kruijf, and Arkaprava Basu for providing valuable comments. Thanks to Shuou Nomura for many detailed comments on the paper and in instigating our development of the mathematical analysis to answer Aged-SDMR's defect quality. Support for this research was provided by NSF under the following grants: CNS-1117782.

9 References

[1] Body biasing. In *S. Narendra and A. Chandrakasan, editors, Leakage in Nanometer CMOS Technologies. Springer US, 2006.*

[2] Semiconductor Industry Association (SIA), Design, International Roadmap for Semiconductors, 2009 edition.

[3] M. Agarwal, B. Paul, M. Zhang, and S. Mitra. Circuit failure prediction and its application to transistor aging. In *VTS '07*.

[4] A. Ansari, S. Feng, S. Gupta, and S. Mahlke. Necromancer: enhancing system throughput by animating dead cores. In *ISCA '10*.

[5] T. M. Austin. Diva: A reliable substrate for deep submicron microarchitecture design. In *MICRO '99*.

[6] J. Blome, S. Feng, S. Gupta, and S. Mahlke. Self-calibrating online wearout detection. In *MICRO '07*.

[7] T. Boon-Chan, J. Sartori, P. Gupta, and R. Kumar. On the efficacy of nbti mitigation techniques. In *DATE '12*.

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

[9] K. Bowman, J. Tschanz, C. Wilkerson, S. Lu, T. Karnik, V. De, and S. Borkar. Circuit techniques for dynamic variation tolerance. In *DAC '09*.

[10] M. A. Breuer, S. K. Gupta, and T. Mak. Defect and Error Tolerance in the Presence of Massive Numbers of Defects. *IEEE Design and Test*, 21(3):216–227, 2004.

[11] A. Calimera, E. Macii, and M. Poncino. Nbti-aware power gating for concurrent leakage and aging optimization. In *ISLPED '09*.

[12] X. Chen, Y. Wang, Y. Cao, Y. Ma, and H. Yang. Variation-aware supply voltage assignment for minimizing circuit degradation and leakage. In *ISLPED '09*.

[13] K. Constantinides, O. Mutlu, T. Austin, and V. Bertacco. Software-based online detection of hardware defects mechanisms, architectural support, and evaluation. In *MICRO '07*.

[14] B. D. Cory, R. Kapur, and B. Underwood. Speed binning with path delay test in 150-nm technology. *Design & Test of Computers, IEEE*, 2003.

[15] J. Donald and M. Martonosi. Power efficiency for variation-tolerant multicore processors. In *ISLPED '06*.

[16] Ernst et al. Razor: A low-power pipeline based on circuit-level timing speculation. In *MICRO '03*.

[17] S. Feng, S. Gupta, A. Ansari, and S. Mahlke. Shoestring: Probabilistic soft-error reliability on the cheap. In *ASPLOS-15*, 2010.

[18] V. Gherman, J. Massas, S. Evain, S. Chevobbe, and Y. Bonhomme. Error prediction based on concurrent self-test and reduced slack time. In *DATE '11*.

[19] D. Gizopoulos, M. Psarakis, S. V. Adve, P. Ramachandran, S. K. S. Hari, D. Sorin, A. Meixner, A. Biswas, and X. Vera. Architectures for online error detection and recovery in multicore processors. In *DATE '11*.

[20] M. Gomaa, C. Scarbrough, T. N. Vijaykumar, and I. Pomeranz. Transient-fault recovery for chip multiprocessors. In *ISCA '03*.

[21] B. Greskamp, L. Wan, U. R. Karpuzcu, J. J. Cook, J. Torrellas, D. Chen, and C. Zilles. Blueshift: Designing processors for timing speculation from the ground up. In *HPCA '09*.

[22] S. Gupta, S. Feng, A. Ansari, J. Blome, and S. Mahlke. The stagenet fabric for constructing resilient multicore systems. In *MICRO '08*.

[23] A. Haggag, M. Moosa, N. Liu, D. Burnett, G. Abeln, M. Kuffler, K. Forbes, P. Schani, M. Shroff, M. Hall, C. Paquette, G. Anderson, D. Pan, K. Cox, J. Higman, M. Mendicino, and S. Venkatesan. Realistic Projections of Product Fails from NBTI and TDDB. In *Reliability Physics Symposium Proceedings*, pages 541 –544, 2006.

[24] T. Hosoi, P. Lo Re, Y. Kamakura, and K. Taniguchi. A new model of time evolution of gate leakage current after soft breakdown in ultra-thin gate oxides. In *IEDM '02*.

[25] U. R. Karpuzcu, B. Greskamp, and J. Torrellas. The bubblewrap many-core: popping cores for sequential acceleration. In *MICRO 42*.

[26] U. R. Karpuzcu, B. Greskamp, and J. Torrellas. The BubbleWrap many-core: popping cores for sequential acceleration. In *MICRO '09*.

[27] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar. Adaptive techniques for overcoming performance degradation due to aging in digital circuits. In *ASP-DAC '09*.

[28] C. LaFrieda, E. Ipek, J. F. Martinez, and R. Manohar. Utilizing dynamically coupled cores to form a resilient chip multiprocessor. In *DSN '07*.

[29] J. Lee and N. S. Kim. Optimizing throughput of power- and thermal-constrained multicore processors using dvfs and per-core power-gating. In *DAC '09*.

[30] X. Li and D. Yeung. Application-level correctness and its impact on fault tolerance. In *HPCA '07*.

[31] Li et al. Understanding the propagation of hard errors to software and implications for resilient system design. In *ASPLOS '08*.

[32] B. Linder and J. Stathis. Statistics of progressive breakdown in ultra-thin oxides. *Microelectronic Engineering*, 72:24 – 28, 2004.

[33] E. J. McCluskey, A. Al-Yamani, J. C.-M. Li, C.-W. Tseng, E. Volkerink, F.-F. Ferhani, E. Li, and S. Mitra. Elf-murphy data on defects and test sets. *VTS '04*.

[34] A. Meixner, M. E. Bauer, and D. J. Sorin. Argus: Low-cost, comprehensive error detection in simple cores. In *MICRO '07*.

[35] S. S. Mukherjee, M. Kontz, and S. K. Reinhardt. Detailed design and evaluation of redundant multithreading alternatives. In *ISCA '02*.

[36] J. Nakano, P. Montesinos, K. Gharachorloo, and J.Torrellas. Revivei/o: efficient handling of i/o in highly-available rollback-recovery servers. In *HPCA '06*.

[37] S. Nomura, M. D. Sinclair, C.-H. Ho, V. Govindaraju, M. de Kruijf, and K. Sankaralingam. Sampling+ dmr: practical and low-overhead permanent fault detection. In *ISCA '11*.

[38] OpenRISC, http://opencores.org/or1k/.

[39] J. Park and J. Abraham. An aging-aware flip-flop design based on accurate, run-time failure prediction. In *VTS '12*.

[40] I. Pomeranz and S. M. Reddy. An efficient non-enumerative method to estimate path delay fault coverage. In *ICCAD '92*.

[41] M. D. Powell, A. Biswas, S. Gupta, and S. S. Mukherjee. Architectural core salvaging in a multi-core processor for hard-error tolerance. In *ISCA '09*.

[42] M. Prvulovic, Z. Zhang, and J. Torrellas. Revive: cost-effective architectural support for rollback recovery in shared-memory multiprocessors. In *ISCA '02*.

[43] J. Rearick. Too much delay fault coverage is a bad thing. In *ITC '01*.

[44] Sampson et al. Enerj: Approximate data types for safe and general low-power computation. In *PLDI '11*.

[45] D. Schroder. Negative bias temperature instability: What do we understand? *Microelectronics Reliability*, 47(6):841–852, 2007.

[46] E. Schuchman and T. N. Vijaykumar. BlackJack: Hard Error Detection with Redundant Threads on SMT. In *DSN '07*.

[47] S. Shyam, K. Constantinides, S. Phadke, V. Bertacco, and T. Austin. Ultra low-cost defect protection for microprocessor pipelines. In *ASPLOS '06*.

[48] T. Siddiqua. *A Multi-Level Approach to NBTI Mitigation in Processors*. PhD thesis, University of Virginia, December 2012.

[49] J. C. Smolens, B. T. Gold, J. C. Hoe, B. Falsafi, and K. Mai. Detecting emerging wearout faults. In *Proc. of Workshop on SELSE*, 2007.

[50] D. J. Sorin, M. M. K. Martin, M. D. Hill, and D. A. Wood. Safetynet: improving the availability of shared memory multiprocessors with global checkpoint/recovery. In *ISCA '02*.

[51] J. Srinivasan, S. Adve, P. Bose, J. Rivers, and C. Hu. Ramp: A model for reliability aware microprocessor design. *IBM, Poughkeepsie, NY*, 2003.

[52] A. W. Strong, E. Y. Wu, R.-P. Vollertsen, J. Sune, G. La Rosa, T. D. Sullivan, and S. E. Rauch III. *Reliability wearout mechanisms in advanced CMOS technologies*, volume 12. Wiley-IEEE Press, 2009.

[53] D. Sylvester, D. Blaauw, and E. Karl. Elastic: An adaptive self-healing architecture for unpredictable silicon. *IEEE Design and Test*, 23(6):484–490, 2006.

[54] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. Mitigating parameter variation with dynamic fine-grain body biasing. In *MICRO '07*.

[55] A. Tiwari and J. Torrellas. Facelift: Hiding and slowing down aging in multicores. In *MICRO '08*.

[56] Tschanz et al. Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In *Symposium on VLSI Circuits*, pages 112–113. IEEE, 2009.

[57] Tschanz et al. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage, ISSCC '02.

[58] B. Tudor, J. Wang, W. Liu, and H. Elhak. Mos device aging analysis with hspice and customsim. Technical report, 2012.

[59] G. Upasani, X. Vera, and A. González. Setting an error detection infrastructure with low cost acoustic wave detectors. In *ISCA '12*.

[60] X. Wang, D. Tran, S. George, L. Winemberg, N. Ahmed, S. Palosh, A. Dobin, and M. Tehranipoor. Radic: A standard-cell-based sensor for on-chip aging and flip-flop metastability measurements. In *ITC '12*.

[61] P. M. Wells, K. Chakraborty, and G. S. Sohi. Mixed-mode multicore reliability. In *ASPLOS '09*.

[62] B. Zandian, W. Dweik, S. H. Kang, T. Punihaole, and M. Annavaram. Wearmon: Reliability monitoring using adaptive critical path testing. In *DSN '12*.