Use It or Lose It: Proactive, Deterministic Longevity in Future Chip Multiprocessors

HYUNGJUN KIM and SIVA BHANU KRISHNA BOGA, Texas A&M University
ARSENIY VITKOVSKIY, Cyprus University of Technology
STAVROS HADJITHEOPHANOUS, University of Cyprus
PAUL V. GRATZ, Texas A&M University
VASSOS SOTERIOU, Cyprus University of Technology
MARIA K. MICHAEL, University of Cyprus

Moore's Law scaling continues to yield higher transistor density with each succeeding process generation, leading to today's many-core chip multiprocessors (CMPs) with tens or even hundreds of interconnected cores or tiles. Unfortunately, deep submicron CMOS process technology is marred by increasing susceptibility to wear. Prolonged operational stress gives rise to accelerated wearout and failure due to several physical failure mechanisms, including hot-carrier injection (HCI) and negative-bias temperature instability (NBTI). Each failure mechanism correlates with different usage-based stresses, all of which can eventually generate permanent faults. While the wearout of an individual core in many-core CMPs may not necessarily be catastrophic, a single fault in the interprocessor network-on-chip (NoC) fabric could render the entire chip useless, as it could lead to protocol-level deadlocks, or even partition away vital components such as the memory controller or other critical I/O. In this article, we study HCI- and NBTI-induced wear due to actual stresses caused by real workloads, applied onto the interconnect microarchitecture and develop a critical path model for NBTI-induced wearout. A key finding of this modeling is that, counter to prevailing wisdom, wearout in the CMP's on-chip interconnect is correlated with lack of load observed in the NoC routers rather than high load. We then develop a novel wearout-decelerating scheme in which routers under low load have their wear-sensitive components exercised without significantly impacting cycle time, pipeline depth, area, or power consumption of the overall router. A novel deterministic approach is proposed for the generation of appropriate exercise-mode data, ensuring design parameter targets are met. We subsequently show that the proposed design yields an $\sim$2,300$\times$ decrease in the rate of wear.

Categories and Subject Descriptors: B.4.3 [**Input/Output and Data Communications**]: Interconnections (Subsystems); B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance

General Terms: Reliability, Design

Additional Key Words and Phrases: Negative-bias temperature instability (NBTI), hot-carrier injection (HCI), wearout, network-on-chip, reliability, lifetime

ACM Reference Format:
Hyungjun Kim, Siva Bhanu Krishna Boga, Arseniy Vitkovskiy, Stavros Hadjitheophanous, Paul V. Gratz, Vassos Soteriou, and Maria K. Michael. 2015. Use it or lose it: Proactive, deterministic longevity in future chip multiprocessors. ACM Trans. Des. Autom. Electron. Syst. 20, 4, Article 65 (September 2015), 26 pages.
DOI: http://dx.doi.org/10.1145/2770873

This work falls under the Cyprus Research Promotion Foundation's Framework Programme for Research, Technological Development and Innovation 2009-10 (DESMH 2009-10), co-funded by the Republic of Cyprus and the European Regional Development Fund, and specifically under Grant IPE/DIEJNHS/S-TOQOS/0311/06.
Authors' addresses: H. Kim, S. B. K. Boga, and P. V. Gratz (corresponding author), Department of Electrical and Computer Engineering, Texas A&M University; A. Vitkovskiy and V. Soteriou, Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology; S. Hadjitheophanous and M. K. Michael, Department of Electrical and Computer Engineering, The University of Cyprus & KIOS Research Center; corresponding author's email: pgratz@gratz.com.

DOI: http://dx.doi.org/10.1145/2770873

1. INTRODUCTION

The continuous aggressive miniaturization of CMOS feature sizes and the resulting increase in transistor density has recently sparked the multicore era. Architects have harnessed this increasing supply of transistors, resulting in the design of parallel systems, including chip multiprocessors (CMPs) [Howard et al. 2011]. In these systems, the on-chip interconnect, typically organized as a network-on-chip (NoC) [Dally and Towles 2001], plays a vital role in enabling communication among the various on-chip computational, memory, and peripheral components, as illustrated in Figure 1. Unfortunately, deep submicron CMOS process technology is marred by increasing susceptibility to wearout, dramatically shortening the useful lifespan of such on-chip parallel systems. Recent ITRS reports indicate a ten-fold decrease in wear-rate will be required to maintain current design lifetimes without dramatically increasing timing margins [ITRS 2009]. As we will illustrate, wearout does not affect all components equally: wear of the cores can often be managed, while wear of the NoC interconnect can be catastrophic. Furthermore, we will show that wear in the NoC is highly dependent upon the actual operational stresses caused by real CMP workloads. In this work, we develop techniques to proactively maintain the CMP's NoC in the face of workload-dependent wear, and hence improve the overall functional lifetime of the CMP as a whole.

Two key operational stress-induced wear mechanisms in current and future CMOS technology are hot-carrier injection (HCI) and negative-bias temperature instability (NBTI) [Oboril and Tahoori 2012]. Both HCI and NBTI cause a shift of the transistor's threshold voltage, leading to switching delay and critical path degradation [JEDEC 2011]. Though these effects do not result in circuit opens or shorts, over time they can cause critical path timing violations. Given equivalent supply voltage and temperature, HCI and NBTI degradation are primarily dependent upon the time transistors have been operating under stress. These types of stresses are essentially data- and usage-dependent in terms of the activity factor (i.e., the fraction of cycles during which a transistor switches) and duty cycle (i.e., the percentage of time the gate's voltage is held at a constant zero), respectively, of the gates in typical CMOS logic circuits.

Figure 1 illustrates a CMP exposed to wearout failures in its various components. As prior work would indicate, individual core wearout and failure need not be catastrophic to the functionality of many-core CMPs due to the inherent core redundancy that a CMP implies [Karpuzcu et al. 2009; Blome et al. 2007; Smolens et al. 2007; Powell et al. 2009; Li et al. 2008a; Huang and Xu 2010]. With increasing numbers of cores, an equivalently smaller portion of the overall system's required throughput is dependent upon each individual core. The component failure scenario (1) of Figure 1 shows this case. Failure caused by wearout of some cores need not result in full-system failure. Instead, the system could suffer some performance loss while preserving correct functionality, assuming core-level error detection and appropriate system support is available [Blome et al. 2007; Smolens et al. 2007; Powell et al. 2009; Li et al. 2008a; Huang and Xu 2010; Skitsas et al. 2013].

For the NoC interconnecting the cores, however, redundancy-based wear resilience breaks down (c.f., component failure scenarios (2), (3), and (4) of Figure 1). Scenario (2) illustrates the case in which a wearout-induced link failure precludes access to a key I/O peripheral, while in scenario (3), link and router wearout has partitioned away a large fraction of the network, making those cores and I/O components inaccessible from the rest of the system. In both cases, wearout is catastrophic in that the system will likely be rendered unusable due to these failures, unlike the core wearout in scenario (1) discussed earlier. Even scenario (4), in which a single link is broken due to wear-induced failure, might lead to a communication protocol-induced deadlock(s), or subnetwork isolation, if the network is not provisioned to address wear-induced failures.

Prior work has proposed various fault-tolerant routing algorithms and fault-insensitive router and link designs in an attempt to manage faults as they occur [Zhang et al. 2008; Schonwald et al. 2007; Fick et al. 2009; Bhardwaj et al. 2012, 2013; DeOrio et al. 2011]; however, network isolation and key resource partitioning

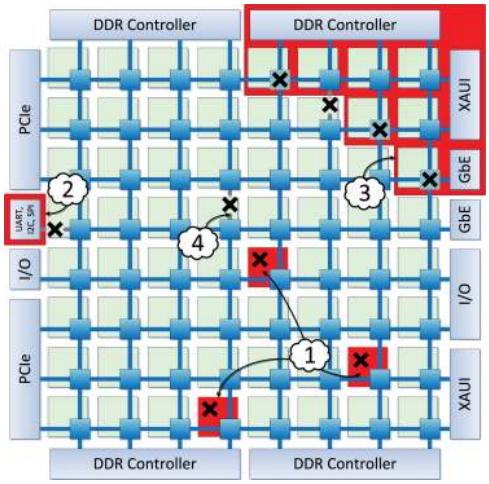


Fig. 1. A 64-core CMP with an 8×8 2D mesh NoC. Components marked with a black × illustrate wearout failure. The failure scenarios are as follows: (1) failure of cores; (2) peripheral device disconnected from the system due to link failure; (3) network segmentation resulting in a disconnected subnetwork; (4) individual link failure.

cannot be fully resolved using only such *reactive* techniques. Ideally, one would prefer to develop *proactive* mechanisms to extend the healthy status of the system without failure rather than react to the faults once they occur. Such proactive mechanisms could be coupled to reactive mechanisms in the hope that the latter would be required less frequently as faults in the system would occur sporadically.

In this work, we present a proactive technique designed to decelerate the effects of aging in the CMP NoC. Based upon detailed HCI and NBTI transistor-level aging models, we develop a novel, critical path-based model to characterize the effects of aging-related wear. Based upon this model, we analyze the NoC router microarchitecture to find the paths most susceptible to wearout. Using real workloads from the PARSEC Benchmark Suite [Bienia et al. 2008], we characterize various wearout mechanisms that map onto those paths. Finally, we develop a wearout-resistant router micro-architecture, coupled with a deterministic generation of exercise mode data, which prolongs circuit lifetime with negligible influence on the router's timing, pipeline, CMOS area requirements, and power consumption. This proposed technique yields a ∼2300× decrease in the CMP wear rate.

This article provides the following contributions.

(1) We present a generalized, microarchitecture-level (rather than device-level) wearout model.
(2) We characterize NoC router and link wearout due to HCI and NBTI under real workloads from the PARSEC Benchmark Suite [Bienia et al. 2008].
(3) We provide a novel wear-resistant router microarchitecture which dramatically improves interconnect lifetime, and hence full-system survivability in the presence of both HCI- and NBTI-wearout mechanisms.
(4) We use a novel systematic approach, inspired by recent work in automatic test pattern generation, to generate exercise mode data which supervises the NoC's lifetime extension, while maintaining a small hardware overhead for the underlying router microarchitecture.

This article is organized as follows. Section 2 examines existing transistor-level models for HCI- and NBTI-induced wear. Next, Section 3 examines the sensitivity of the router's critical path to wear by analyzing the activity and duty cycle of its critical path, and characterizes the router's wear caused by the behavior of real application

workloads. Based upon these wearout models, Section 4 then develops a circuit path delay model for workload stress-induced wear. Section 5 proposes a novel router microarchitecture to improve the lifetime of NoC routers under realistic workloads which utilizes the exercised mode data derived by the systematic methodology described in Section 6, while Section 7 evaluates the proposed design. Finally, Section 8 presents prior related work, while Section 9 concludes article.

2. BACKGROUND

Prior research shows that the two dominant CMOS transistor physical failure mechanisms are hot-carrier injection (HCI) and negative-bias temperature instability (NBTI) [Nassif et al. 2007]. Under both, failure mechanisms charge becomes trapped in or near the gate oxide, resulting in a slow increase of the transistor's threshold voltage (V_{th}). This in turn causes the delay in transistor state switching to expand.

In traditional circuit CMOS design, the clock frequency of a given design is determined by the circuit path which exhibits the longest latency between its end latches. This *critical path* comprises a chain of connected gates between latches. As HCI- and NBTI-induced aging progresses, it gradually extends the delay of each gate found in this chain, slowing down the entire critical path. In modern CMOS designs, due to this age-induced slow-down and other causes, such as process variation [Kuhn 2007], designs are given timing guard-bands so as to guarantee their intended functionality for a certain duration of time [Agarwal et al. 2007]. Once the aggregate increase in delay along a timing-critical path exceeds this guard-band, due to the aggregation of increasing delays occurring in individual gates along this path, the functionality of the system is no longer assured. The moment at which this timing violation first occurs determines the system's useful life span. Of course, HCI and NBTI impact all transistors in the design (not only those in the critical path); however, those on the critical path are more likely to exceed the guard-band, thus causing a critical failure.

In this section, we first describe the impact of these aging mechanisms upon V_{th} using transistor-level analytical models. We then examine a number of specific NoC router critical paths which are most susceptible to these aging effects, since they determine the system's clock rate.

2.1. Failure Mechanisms

Design rules and operating conditions are precisely chosen to ensure correct product functional operation over its intended lifetime [Li et al. 2008b]. To obtain a given level of performance when utilizing an integrated circuit under various design constraints, it becomes imperative to create and analyze the reliability model of the digital system under consideration and during its design phase.

As previously discussed, the HCI and the NBTI mechanisms do not induce failures; rather, they shift parameters over time under circuit operational stresses. The Reaction-Diffusion (R-D) model uses the threshold voltage (V_{th}) shift as a proxy of NBTI and HCI stress [Wang et al. 2011]. A shift in V_{th} causes transistor delay degradation according to the Alpha Power Law [Sakurai and Newton 1990]:

$$d_g \propto \frac{V_{dd}}{\mu(V_{dd} - V_{th})^{\alpha}}, \tag{1}$$

where d_g is the transition delay, $\mu \propto T^{-1.5}$ (T being temperature), and $\alpha = 1.3$.

Lifetime can be defined as the time until an important material of a component or device parameter degrades beyond the point at which the device or circuit can function properly in its originally intended application. For a single gate, when ΔV_{th} reaches some level (in practice, it is usually 10% [Wang et al. 2011]), the transistor is considered to be over-aged. For the multigate path, the cumulative transistor delay shift increases faster than a single gate's worst-case delay degradation. Therefore, a total gate delay shift over the entire path, when its value reaches or exceeds the 10% threshold mark, can serve as a lifetime period indicator.

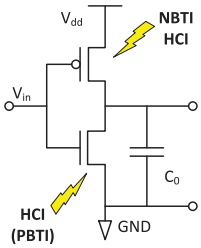
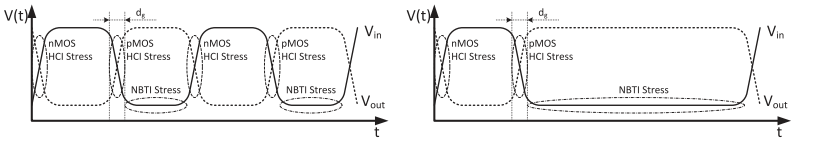


Fig. 2. CMOS inverter.



(a) Higher switching activity, lower duty cycle.

(b) Lower switching activity, higher duty cycle.

Fig. 3. HCI and NBTI stress time windows for a CMOS inverter.

Figure 2 shows a typical CMOS inverter, indicating the failure mechanisms associated with each type of transistor: HCI affects both the nMOSFET and pMOSFET transistors, while NBTI affects only the pMOSFET transistor (note that PBTI is the complement of NBTI and affects nMOSFET transistors only; however, its effect is generally considered to be much smaller than that of NBTI). The following sections present a device parameter degradation model that captures the HCI and NBTI wearout effects.

2.1.1. Hot-Carrier Injection (HCI). Hot-carrier injection (HCI) is a wear-out mechanism which occurs when carriers flow along the channel in MOSFET transistors and gain sufficient kinetic energy to be injected into the gate oxide resulting in a charge trap and interface state generation. This leads to a gradual transistor parameter shifting, including switching frequency degradation, rather than causing an immediate failure event [JEDEC 2011].

A substrate current-based (I_{sub}) model is commonly used to estimate HCI's effect. Prior work shows that the threshold voltage shift due to HCI under DC stress is

$$\Delta V_{th\_HCI}|_{DC} = A(I_{sub})^m t_{stress}^{n'}, \tag{2}$$

where A is the material-dependent parameter, t_{stress} is the stress time, and n' and m are technology-related exponents [Li et al. 2008b; JEDEC 2011].

According to the Alpha Power Law of Equation (1), the delay of a transistor depends linearly on threshold voltage for small shifts, so the gate delay shift can be expressed as

$$\Delta d_{g\_HCI}\big|_{DC} = \hat{A}(I_{sub})^m t_{stress}^{n'}, \tag{3}$$

where $\hat{A}$ is a fitting constant.

The lifetime of a device exposed by a direct HCI effect [JEDEC 2011] is

$$TTF_{HCI}|_{DC} = A_{HCI}\,(I_{sub})^{-N'}\, e^{\left(\frac{E_{aHCI}}{kT}\right)}, \tag{4}$$

where E_{HCI} is the apparent activation energy, I_{sub} is the substrate current under stress at $V_G = V_D$, T is the runtime temperature, k is Boltzmann's constant, N' is the technology-related exponent, and A_{HCI} is a fitting constant.

HCI stresses the device only during dynamic transitions when current flows through the device. Figure 3 shows voltage waveforms of a standard CMOS inverter. The pMOSFET transistor suffers HCI stress when the output of the inverter is pulling up and C_0 is charging up (see Figure 2). The nMOSFET transistor experiences HCI degradation during the reverse dynamic stage, when the output of the inverter is discharged to the ground (low-voltage level) [Li et al. 2008b]. Thus, each of the CMOS transistors experiences degradation only during half of a switching period, and hence the relation between HCI stress time t_{stress}^{HCI} and runtime t can be derived as

$$t_{HCI\_stress} = d_g\, f\, \alpha_{SA} t, \tag{5}$$

where d_g is the transition delay, α_{SA} is the switching activity, and f is the frequency.

Since HCI stress occurs during the device turn-on and turn-off periods, the impact of HCI under AC stress can be extracted from Equations (3) and (4) using (5):

$$\Delta d_{g\_HCI}\big|_{AC} = A(I_{sub})^m (d_g f \alpha_{SA} t)^{n'}. \tag{6}$$

And finally, the last equation is transformed into a relation for HCI lifetime:

$$TTF_{HCI}(T, \alpha_{SA})\big|_{AC} = A_{HCI} \frac{1}{d_g f \alpha_{SA}} (I_{sub})^{-N'} e^{\left(\frac{E_{aHCI}}{kT}\right)}. \tag{7}$$

This relation shows that the lifetime of a transistor due to *HCI* is inversely related to the switching activity α_{SA} of the gate input. Hence, frequent switching, such as that shown in Figure 3(a), not only increases the dynamic power consumption but also speeds up the aging effect, whereas a gate with less frequently-occurring transitions, as shown in Figure 3(b), will experience lighter HCI-induced aging.

2.1.2. Negative-Bias Temperature Instability (NBTI). Negative-bias temperature instability (NBTI) is a wear-out effect that influences pMOSFET transistors as long as they operate in inversion (i.e., a "0" voltage on the input of an inverter, as shown in Figure 2). Thus the data-dependent stress caused by NBTI is very different from that of HCI, which is under stress during voltage-level switching. NBTI changes the pMOSFET transistor parameters over time. In particular, it leads to an increase in the threshold voltage (V_{th}) as well as a reduction in the drive current due to charge carrier mobility degradation. As with HCI, NBTI does not result in complete circuit failure but rather in circuit-speed degradation. JEDEC reports that process technology scaling will lead to a larger NBTI-induced threshold voltage in pMOSFET transistors [JEDEC 2011]. It has been reported that, unlike HCI, some degree of recovery from NBTI degradation can occur in the event that a relaxation period occurs after the stress period [JEDEC 2011; Li et al. 2008b; Wang et al. 2011].

We use the AC stress model for NBTI degradation under high-frequency CMOS operation, as proposed by Lu et al. [2009], which provides a theoretical upper-bound estimation of the NBTI effect in terms of time as

$$\Delta V_{th\_NBTI} = A\left(\frac{\beta}{1-\beta}\right)^n t^n e^{\left(-\frac{nE_{aNBTI}}{kT}\right)}, \tag{8}$$

where β is the duty cycle (i.e., the fraction of time the gate's voltage is zero), E_{aNBTI} is the activation energy, T is the temperature, t is the operating time, k is Boltzmann's constant, n is the time exponent, and A is a fitting constant [Lu et al. 2009].

According to the Alpha Power Law (1), the first-order gate delay can be approximated as a linear function of the threshold voltage. Hence, the gate delay shift can be expressed as

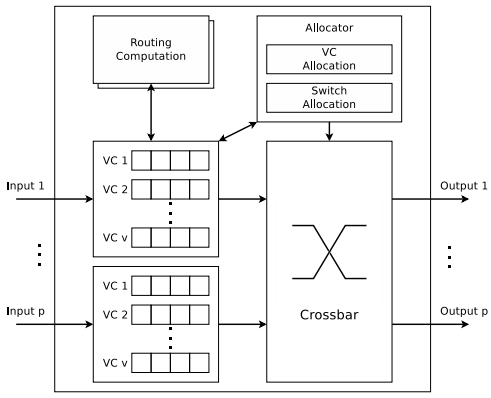
$$\Delta d_{g\_NBTI} = \hat{A}\left(\frac{\beta}{1-\beta}\right)^n t^n e^{\left(-\frac{nE_{aNBTI}}{kT}\right)}. \tag{9}$$

The lifetime of a single transistor under AC stress can be derived from (9) as

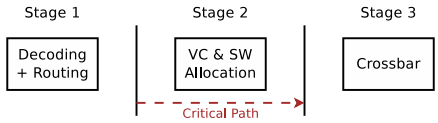
$$TTF_{NBTI} = \left[A_{NBTI}\left(\frac{1-\beta}{\beta}\right)^n e^{\left(\frac{nE_{aNBTI}}{kT}\right)}\right]^{1/n}. \tag{10}$$

Thus, lifetime degradation due to NBTI depends on the duty cycle of the input signal. Transistors with a smaller duty cycle, such as the duty cycle shown in Figure 3(a) in comparison to the duty cycle shown in Figure 3(b), experience a slower degradation rate.

2.1.3. HCI and NBTI Failure Mechanism Analysis. It may first appear that a technique which improves device lifetime by decreasing NBTI must come at the cost of a comparable degradation caused by HCI-related wear (and vice versa). We note, however, that the activity factor α_{SA} is not the inverse of duty cycle β; when β is large, it is



(a) Router block diagram.



(b) Router pipeline stages.

Fig. 4. Baseline router.

possible to make a substantial change to β without proportionally impacting α_{SA}. Furthermore, because of the $\frac{1}{(1-\beta)}$ term in Equations (8) and (9), large βs tend to have a disproportionate impact on aging-related slow-down. Even a small improvement in the value of β can therefore have a substantial positive effect on the overall device lifetime (especially when β is relatively large).

2.2. Router Microarchitecture

The canonical NoC virtual channel router is proposed by Peh and Dally [2001]. Its block diagram is shown in Figure 4(a). The major building blocks of this NoC router are input channels, a crossbar (switch), and the control logic which includes the switch and virtual channel allocators. When used in a two-dimensional mesh NoC architecture, typically five input and output channels, p, are used to connect its four immediate neighbors at the cardinal points and its local processing element. An input channel is composed of a given number of virtual channels (VCs), each of which includes registers to keep track of their statuses and buffers to store flits (flow-control digits, a logical fixed-segment of a packet). The routing units also examine flits found in the input channels to determine the next-hop direction packets should take (i.e., east, west, north, or south directions). The VC allocator assigns a free VC at a downstream router to a head flit, the first flit of a packet. If the head flit successfully obtains a VC, it competes with any other flits destined to the same output port during switch allocation. Body and tail flits in the same packet skip the routing and VC allocation stages and directly proceed to the switch allocation stage. Once switch allocation is complete, the flit traverses the crossbar.

Our baseline router performs both VC and switch allocation during the same cycle by speculatively allowing a packet to compete for the switch while it is still competing for a free VC at the downstream router [Peh and Dally 2001]. Figure 4(b) shows the baseline router pipeline. Flit decoding and routing computation are carried out in Stage 1. The combined VC and switch (SW) allocations are carried out in Stage 2. In Stage 3, flits traverse the crossbar.

NBTI and HCI both slow transistor switching, and thus damage from the circuit aging first takes place where timing is most critical. To determine the critical path of the baseline router, we adapted our baseline RTL from the publicly available router RTL model designed by Becker [2012]. This RTL model is synthesized using Synopsys Design Compiler and mapped to TSMC 45nm standard cell library to a 1GHz frequency. All critical paths with slack less than 10% of the clock frequency are gathered using Synopsys Design Vision and analyzed offline.

The results of this analysis are highlighted in Figure 4(b). We find that all timing critical paths (i.e., those within 10% of the 1GHz clock frequency) pass through the VC and switch (SW) allocators. These results correspond well with prior research [Peh and Dally 2001].[1] The utilization of the allocators is closely related to the router's incoming rate or the number of flits traversing the given router per cycle, because the allocators are enabled by the input channel which sends the request signal to the allocators when it has flits to forward. As the critical path is initiated by the request signal, the wire activity along the path is dependent upon how often the request signal is set, which in turn is determined by the workload's utilization of that router. We therefore expect that the stress time for HCI and NBTI, which are closely related to the activity factor and the duty cycle, thereby should be also closely correlated to the router's utilization. Next, in Section 3.3, we perform an in-depth study on the impact of typical CMP workloads on the router's critical path in terms of activity factor and duty cycle.

3. ROUTER WEAROUT CHARACTERIZATION

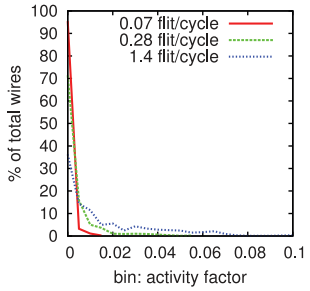
To examine the sensitivity of the router's critical path to HCI and NBTI wear, we first analyze the activity of wires residing in the baseline router under synthetic workloads. The router has five physical channels, four VCs per physical channel, and 4 flit-deep buffers per VC. Dimension-order routing (DOR) is used. The network is designed to transfer 64-byte memory blocks where the link-width between adjacent routers is 128 bits, discounting any flow-control signals. Hence, if a packet includes such data, it is composed of 5 flits (1 head flit containing routing information and metadata and 4 data flits); otherwise, it is composed of only 1 head flit. The workload is generated with an arbitrary injection rate, maintaining a 50-50% proportion of 1-flit and 5-flit packets. As described in Section 2.2, all paths from the post-synthesis router model with 10% or less slack relative to the 1GHz clock frequency, were examined. In particular, information about the activity factor and duty cycle of all wire nodes along each critical path under these workloads was retained for analysis.

3.1. Impact of Workload upon Router Activity Factor

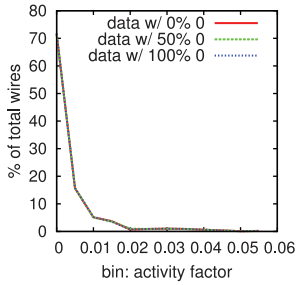
HCI is proportional to the activity factor of the NoC's interconnect wires such that a higher activity factor results in an accelerated (higher) aging rate (refer to Section 2.1.1). Figure 5(a) shows the histogram of activity factors of the wires on the critical paths of an NoC router with respect to varying incoming rates. The first observation we make is that the nodes have a quite low activity factor, the vast majority switching less than 10% of the time (activity factor of 0.1).

Intuitively, the higher incoming rate should cause a correspondingly higher activity factor. This implies that a router experiencing traffic from an application that injects more frequently ages at a faster rate because of HCI stress. Hence, it is desirable to keep a low incoming rate so as to improve the longevity of the router. We find, however, the activity factor does not increase very significantly as the incoming rate increases. For instance, even at the very high incoming rate of 1.0 flits/cycle, the activity factors of most of the wires remain at a relatively low value, and only 7.7% of wires have an activity factor greater than 0.1.

[1]Some prior work highlights the credit return path as the critical path within the router; in our experiments, assuming 6mm links between routers, we found that the credit return path was not found on the critical path. With longer links, however, the return path might become critical, requiring further analysis.



(a) Activity factor with respect to router incoming rate.



(b) Activity factor with respect to data content (the packet's data bits percentage at zero).

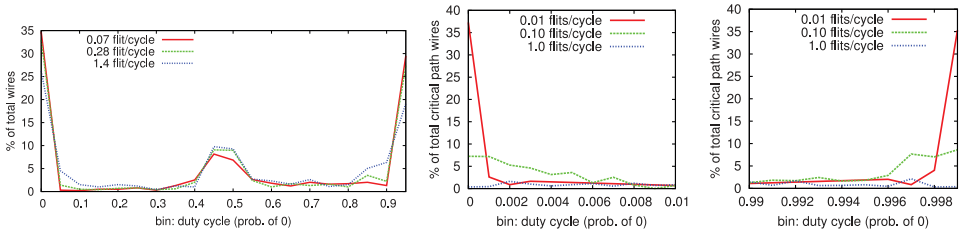
Fig. 5. Sensitivity to the activity factor.

Without a priori knowledge of the router's critical path, one might expect that the content of the data traversing the network would also affect the activity of routers. Figure 5(b) shows the histogram of activity factor with various data contents (percentage of logical zeros in each data-flit vector) at a fixed incoming rate of 0.10 flits/cycle to examine the router's critical path activity factor sensitivity to data content. As expected, the data content does not affect the activity factor of the wires along the router's critical path.

3.2. Impact of Workload upon Router Duty Cycle

NBTI is highly sensitive to the duty cycle of gates (see Section 2.1.2). Figure 6(a) depicts the histogram of wires along the critical path at a given duty cycle for different incoming rates in terms of flits per cycle. In Figure 6, the width of each bin is 0.05; hence bin[0] shows the percentage of gates with a duty cycles in the range of [0, 0.05) and so on. In general, the majority of gates fall into duty cycle bins near 0, 0.5, or 1.0, regardless of the incoming rate. As the incoming rate grows, the bins at the two ends of the spectrum fall while the central part moves up, indicating that an increasing flit incoming rate causes less skew in the duty cycle towards these extremes. Figure 6(b) and Figure 6(c) magnify the two ends of Figure 6(a). To improve observability in these figures, we use a narrower bin width of 0.001. With the increased resolution, we note that higher incoming rates have great impact on duty cycles at these extremes, reducing the percentage of wires along the critical path with the highest and lowest duty cycle from ~35% to near 0%.

The incoming rate's effect on the duty cycle causes notable differences in the NBTI's impact upon gate delay. As shown in Equation (9), there is a nonlinear relationship between duty cycle and gate delay such that the gate delay shoots up as the duty cycle



(a) Range = 0∼1.0, bin width = 0.05.

(b) Range $<$ 0.01, bin width = 0.001.

(c) Range $>$ 0.99, bin width = 0.001.

Fig. 6. Histogram of duty cycle with respect to incoming rate.

Table I. System Setup

Cores	64 on-chip, in-order, Alpha ISA
L1 Cache	32KB instruction/32KB data, 4-way, 64B lines, 3-cycle access time MESI cache coherent protocol
L2 Cache	64-bank fully shared S-NUCA, 16MB, 64B lines, 8-way associative, 8-cycle bank access time
Memory	150-cycle access time, 8 on-chip memory controllers
Network	8 $\times$ 8 Mesh, X-Y routing, 4 VCs/port, packet length: 1 flit or 5 flits

gets closer to 1.0. Hence, for example, it is ideal to have two gates in a given path with the same duty cycle of 0.5, instead of having two gates with duty cycle of 0.0 and 1.0, respectively, in terms of the path-cumulative impact of NBTI upon gate delay. The delay increase due to NBTI from a single gate under a duty cycle of 1.0 alone will greatly exceed the sum of delay increases from two gates, each with a duty cycle of 0.5. Thus, it is preferable to have a higher incoming rate with more gates with duty cycles closer to 0.5 than to have a lower incoming rate and gates that have duty cycles closer to 1.0; although it is notably counterintuitive that accelerated wear-out occurs when routers are underutilized. Based upon these observations, we will next develop a multigate delay model in Section 4.

We examined the router's critical paths to determine why these paths exhibited such a skewed duty cycle and low activity factor (see Section 3.1). In a router, the longest paths through the crossbar and VC allocators are primarily concerned allocation corner-cases, such as multiple simultaneous incoming packets attempting to allocate a VC with limited available VCs. These cases are relatively rare, only occurring under highly loaded network conditions, thus these control signals switch infrequently and have very poor duty cycles when the NoC experiences low loads.

3.3. Workload Characterization

Having identified the per-router incoming rate to be a critical workload characteristic that correlates with wear, we now examine the router-to-router incoming rate variance in realistic workloads. In this study, we use the PARSEC benchmark suite as our workload, as these benchmarks mimic a range of representative next-generation, large shared-memory, multithreaded programs for CMPs [Bienia et al. 2008]. The diversity of the PARSEC benchmarks makes them especially useful for this study, as they span a diverse range of emerging applications with varying on-chip communication spatiotemporal characteristics. Specifically, with the PARSEC benchmarks, one observes different and varying behaviors in the NoC's packet (or flit) incoming rate, as will be outlined next in detail.

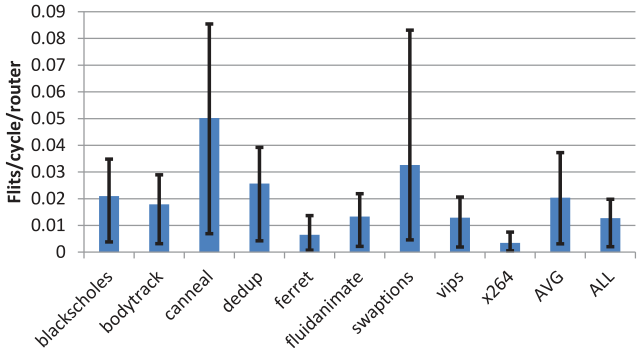


Fig. 7. Average flit incoming rate per router for the entire range of the PARSEC benchmark suite (bars) and related range of incoming values (lines).

The realistic workloads are generated from the gem5 simulator [Binkert et al. 2011] emulating a 64-core system executing multithreaded programs in the PARSEC v2.1 [Bienia et al. 2008] benchmark suite. Table I shows the details of the system setup used in our simulations. We first generate NoC traffic for each application for its region of interest (ROI). We then count the number of flits traversing each router to compute the incoming rate of that router. We note that the term *incoming rate* here is the number of flits injected to a particular router per unit time, rather than the number of flits generated and injected to the network as a whole. This includes the number of flits generated by the router's local processing element and the flits going through or headed for the router. The reason for concentrating on the incoming rate temporal characteristics is that the router's critical path activity is highly related to the frequency of flit arrival (see Sections 3.1 and 3.2).

Figure 7 depicts the average number of flits injected into a router (shown by the solid bars) according to the aforementioned experimental setup, per unit time for each PARSEC benchmark. The incoming rate at routers, on average, is 0.02 flits per cycle. It varies across the programs under examination ranging from 0.003 (*x264*) to 0.05 (*canneal*). The average incoming rate also varies within the same application based on the cartesian location of the router, and the variance is captured by the dark line over each bar. The bottom of the line shows the incoming rate of the router which handles the least traffic among the routers in the network for that benchmark (*min incoming rate*), and the top of the line denotes the incoming rate of the busiest router (*max incoming rate*). Hence, the per-router incoming rate under the PARSEC workloads actually varies between 0.0005 (*min* of *x264*) and 0.085 (*max* of *canneal*). In Figure 7, "AVG" denotes the arithmetic means of the average incoming rate (the bar) and the *min* and *max* incoming rates (the line) across the entire range of benchmarks. "ALL" captures those three incoming rates when the system runs all the benchmarks, one at a time, sequentially.

In general, the average incoming rate seen at each router is quite low, at 0.02 flits per cycle. Hence, HCI-induced aging is not expected to contribute significantly to gate delay under these workloads. Alternately, as discussed previously, a low incoming rate causes NBTI-induced aging. Thus, routers running PARSEC workloads are exposed to accelerated NBTI-induced aging due to their light traffic. Furthermore, there is an observable high variance in their spatially-distributed network flit incoming rates such that some routers executing the *ferret* and *x264* benchmarks experience even less than a 0.001-flits-per-cycle incoming rate. We therefore focus on NBTI aging in the remainder of this study.

4. PATH DELAY

In Section 2.1, we examine existing formulas characterizing wear-induced transistor gate delay. While these equations accurately model the incremental breakdown of

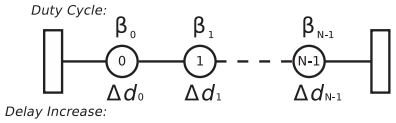


Fig. 8. An example circuit path with multiple gates.

individual gates, we observe that a single gate is only one component of a particular critical path. Here, we derive formulas to compare the relative lifetime of two systems that operate under the same conditions, focusing on the microarchitectural level. There are a number of delay models which take into consideration the aging effect at the transistor level or gate level, but few exist at the microarchitectural level. Ultimately, to calculate the point at which gate delay compromises timing along a particular path, one must examine the cumulative increase in delay (delta-delay) along that path. Here, we propose a method for computing the relative lifetime of a path between latches given the duty cycle of each gate along that path.

We assume that a number of sequential gates comprise a circuit path, as shown in Figure 8. Along this path, the delay increase due to the ith gate with duty cycle β_i at time t can be expressed as

$$\Delta d_i(\beta_i, t) = \psi \times t^n \times \left(\frac{\beta_i}{1-\beta_i}\right)^n, \tag{11}$$

where the constant ψ includes all other terms in Equation (9) under the assumption that those will remain constant under the same condition. The cumulative delay increases along a path with N gates at time t can be computed as

$$\Delta d(t) = \sum_{i=0}^{N-1} \Delta d_i(\beta_i, t) = \psi \times t^n \times \sum_{i=0}^{N-1} \left(\frac{\beta_i}{1-\beta_i}\right)^n. \tag{12}$$

A system is reliable as long as the $\Delta d(t)$ of the critical path is smaller than the guard band. Hence, we define lifetime, $T_{lifetime}$, such that $\Delta d(T_{lifetime}) < guardband$. The *acceleration factor* (AF) is defined as the ratio of the lifetime of the system under consideration, $T_{lifetime}(x)$, and a reference system, $T_{lifetime}(ref)$:

$$AF(x) = \frac{T_{lifetime}(x)}{T_{lifetime}(ref)} = \left(\frac{\sum_{j=0}^{M-1}\left(\frac{\beta_j}{1-\beta_j}\right)^n}{\sum_{i=0}^{N-1}\left(\frac{\beta_i}{1-\beta_i}\right)^n}\right)^{1/n}, \tag{13}$$

where β_i is the duty cycle of the ith gate on the critical path of the system under consideration, and β_j is the duty cycle of the jth gate on the critical path of the reference system. In Equation (13), it is assumed that the number of gates on the critical path of the two systems are N and M, respectively. We note that the method proposed here computes the relative lifetime improvement under NBTI degradation. As discussed in Section 3.3, under the workloads examined, HCI degradation is low and relatively insensitive to the incoming rate; thus we are not modeling its effect on lifetime here.

5. LIFETIME-EXTENDING ROUTER MICROARCHITECTURE

As outlined in Section 3, the aging process is incoming rate-dependent along the critical path. The gate delay increases and the timing constraints are violated along the critical path first. A low incoming rate causes a biased duty cycle in the wires along the critical paths, because those paths deal with allocation corner-cases which are rare,

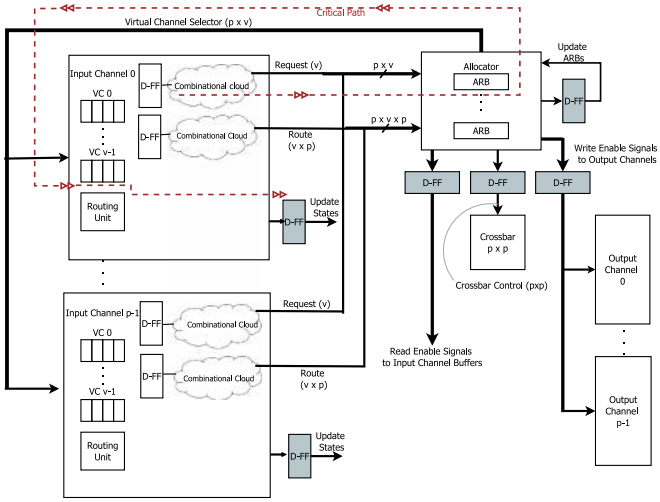


Fig. 9. Virtual channel and switch allocation stages.

unless the load is very high. These biases accelerate NBTI; thus the router requires an increased incoming rate to improve the duty cycle of nodes along critical paths, which subsequently improves lifetime. However, increasing the incoming rate artificially yields other problems, such as increased power consumption. Also, as mentioned in Section 3.1, an increase in incoming rate slightly increases its activity factor as well, which accelerates the HCI effect. The duty cycle must therefore be improved without increasing the activity factor significantly. We note that although duty cycle and activity factor are related, it is possible to reduce the duty cycle of a node substantially without increasing the activity factor substantially by infrequently changing the value of that node. Hence we propose a method to exercise the critical path which improves the duty cycle while minimally disturbing the activity factor, that is, improving NBTI without substantially impacting HCI. The goals of the proposed mechanism are (1) to improve the duty cycle by allowing the circuits to operate at a greater portion of their time in the "1" state without affecting the actual data values being transferred, (2) to not change the state of the router, (3) to not worsen the critical path timing, and (4) to not significantly increase the activity factor.

5.1. Router Critical Path

Figure 9 shows the second pipeline stage of the router (VC and SW allocation) in detail. As indicated by the dotted line in Figure 9, the critical path of the NoC router starts with the flip-flops inside VCs, passes through the allocators and ends again in one of the VCs. Each VC sends a one-bit "Request" signal to the allocator to reserve a VC at the downstream router and/or to bid for switch bandwidth at the crossbar so that the crossbar can be traversed by competing flits. There are p physical channels, each of which has v virtual channels; hence, there are $p \times v$ such control bits in total. Each "Request" signal must be sent with a p bit-width "Route" signal giving the allocator the information as to where the corresponding flit is destined (i.e., to the allocated VC located at a physical port downstream). There is a combinational cloud within each of the Input VCs situated between the flip-flops which reside at the start of the second pipeline stage and the output of the Input channel blocks comprising the "Request" and "Route" signals.

The netlist which represents the combinational logic in a pipeline stage can be represented as a directed acyclic graph (DAG) with a set of primary inputs and a set of primary outputs. All vertices of the graph comprise the gate instances, while the graph edges represent the connections between the gates. A timing arc on this DAG

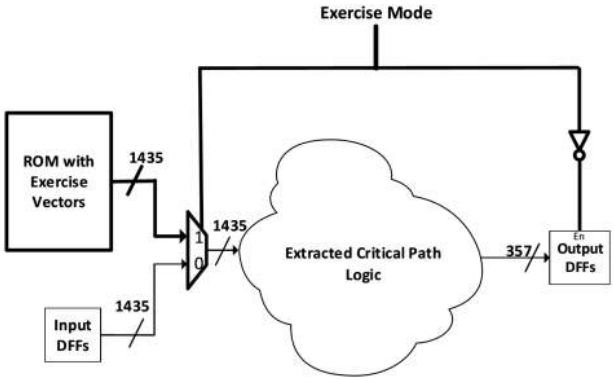


Fig. 10. Critical path logic with proposed exercise logic. Additional exercise logic is shown in a darker shade.

can be defined as a path from any of the primary inputs to any of the primary outputs. By starting at the endpoint of a timing arc and building the logic cone backwards until a set of primary inputs are reached (basically a graph traversal using breadth-first search or depth-first search), all the logic gates which affect that particular path can be extracted. We have constructed such a connectivity graph for our netlist, obtained after synthesizing our baseline router. The critical path logic is extracted by constructing the logic cone for each of the timing paths which have slack of less than 10%.

5.2. Exercise-Mode Logic for Duty-Cycle Balancing

We propose balancing the duty cycle of nodes along the critical paths within the router through the allocators by exercising these paths when the router is quiescent (i.e., there are no packets in-flight through the router). We consider all paths with $\leq$10% slack for this purpose. To ensure that all nodes along critical paths are exercised, we characterize the complete logic circuit which forms each critical path. This critical path logic is extracted from the netlist generated by the the router synthesis, as described in Section 5.1. The resultant combinational logic cloud has 1,435 inputs and 357 outputs.[2]

Figure 10 shows a block diagram of the second pipeline stage which contains the router's critical path with the proposed additional "exercise mode" logic darkened. The exercise mode signal will be high whenever the router is quiescent for a period of time.[3] When the exercise mode signal is high, the input to the critical path logic is taken from the ROM which contains a set of "Exercise vectors" which aim to improve the duty cycle of nodes along the critical paths (the generation of these vectors is described in the next section). As the exercise mode logic must not be allowed to change router state or propagate to the next pipeline stage, the flip-flops or latches between the allocator and the next stage are disabled during the exercise mode.

In order to mitigate the impact on activity factor, the exercise mode input vector from the ROM is rotated within a predefined period. A counter maintains the number of cycles for which exercise mode is active and generates a "toggle" signal (used to change the input vector) once it reaches the predefined rotation period. We note that the duty cycle is insensitive to the frequency of input vector rotation, while the activity factor is linearly related to it. We tested a range of rotation periods, between 16 and 2,048 clock cycles. We explore the implications of period length on the circuit's energy consumption and lifetime in Section 7. In the effort to minimize the impact on the

[2]While the impact of adding a 1,435-bit wide mux could be significant, as we will discuss in Section 6, through vector optimization the overheads can be reduced dramatically.

[3]After experimentation with different values, a period of 16 cycles is chosen to maximize the lifetime gain while minimizing impact on energy.

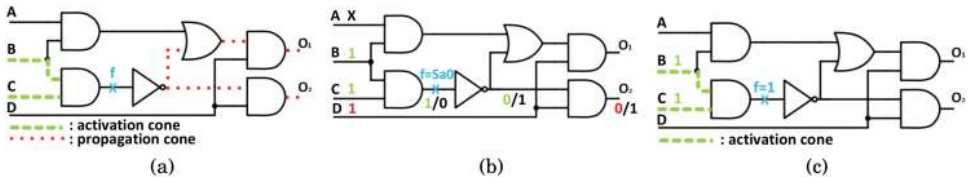


Fig. 11. (a) Activation and propagation cones for fault location f; input signals B, C (A,B,C,D) determine activation (propagation); (b) test generation for f stuck-at-0; $B=1$ and $C=1$ activate the fault and $D=1$ propagates its effect to O_2; possible test vectors $ABCD=X111 = \{0111, 1111\}$; (c) let f be a critical net; exercising $f=1$ requires activation of f stuck-at-0 with $B=1$ and $C=1$.

router's timing and hence the clock rate, vector generation and all other exercise mode selection logic, as shown here, are placed off the critical path.

6. VECTOR GENERATION

As mentioned in Section 5, data is injected during the exercise mode of the router for the purpose of balancing the duty cycle of the nets on the critical paths. In order to optimize this process, we consider deterministic generation of the data to be injected. This particular problem resembles the Automatic Test Pattern Generation (ATPG) process, a well-known NP-complete problem [Abramovici et al. 1990] used for manufacturing tests for integrated circuits [Bushnell and Agrawal 2000]. The ATPG process involves the generation of a set of vectors, called tests, which are applied to each manufactured circuit in order to detect possible defects. ATPG is typically performed at the gate level using predefined fault models, such as the established stuck-at-fault model in which each signal may be stuck to either the logic *"1"* or the logic *"0"* value.

6.1. ATPG Preliminaries and Basic Concepts

The basic ATPG procedure followed in generating a test vector for stuck-at fault tests comprises two phases: the fault activation phase and the fault propagation phase. During fault activation, the fault location (signal) is activated by injecting the opposite of the fault value. The part of the netlist driving the fault location is referred to as the activation cone. The fault propagation phase involves the propagation of the fault effect to some observable output signal. The part of the circuit driven by the fault location is referred to as the propagation cone, and it contains all the possible propagation paths from the fault location to the output signals. Figure 11(a) illustrates the activation and propagation cones for the fault location f in the given netlist.

During ATPG, a signal justification procedure is performed during each of the two phases. Justification during fault activation determines values on the input signals to allow for the activation of the fault, whereas justification during fault propagation determines the values of remaining input signals to allow for fault propagation via some propagation path. Figure 11(b) illustrates one such scenario which sets $B=1$ and $C=1$ during the activation phase for the fault f stuck-at-0, and $D=1$ in order to propagate the fault to the output signal O_2. It is noted that signal A is not set and assumes the "don't care" value (X), which implies that it can be set to any of the two logic values. In this example, if a stuck-at-0 fault exists at f, the value at the output O_2 is '*1*'; otherwise, it is '*0*' (the composite value v_{ff}/v_f stands for fault-free value v_{ff} and faulty value v_f at f). We note that typically the fault propagation phase in ATPG is harder than the activation phase, as it involves the selection of propagation paths and constrained justification based on the results of the activation phase. Nevertheless, both processes are NP-complete due to the justification process which is, in the worst-case, exponential to the number of input signals.

The problem examined in this work resembles an easier, restricted version of the ATPG problem previously discussed. The process of exercising the value '*1*' at some critical net f corresponds to activating the stuck-at-0 fault at f. No propagation is necessary in this case; hence, it suffices to justify the activation value in order to

Possible MUX locations

Vector	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
v_1	X	1	X	X	1	1	1	X	0	X
v_2	0	1	X	1	X	1	1	X	1	X
v_3	1	1	X	X	0	0	1	0	1	X

(a)

MUX locations

Vector	l_1	l_5	l_6	l_9
v_1	X	1	1	0
v_2	0	X	1	1
v_3	1	0	0	1

(b)

Fig. 12. (a) Possible ROM of size 3×10 with 10 possible MUX locations; (b) necessary ROM of size 3×4 with $4 + 4$ MUX locations.

generate the necessary exercise vector. For example, it suffices to set *B=1* and *C=1* in Figure 11(c) in order to exercise signal *f* (which could belong to the critical netlist). The generated vector in this case is $ABCD = \{X11X\}$.

6.2. Hardware Overhead Optimization via Compaction of Exercise Data

A considerable portion of the hardware overhead of the exercise logic, given in Figure 10, consists of the ROM which stores the exercise vectors as well as the various multiplexers (MUXes) inserted to allow for the ROM vectors to be exercised. Both the size of the ROM and the number of new MUXes is data-dependent on both dimensions of the exercised data matrix. To better understand this issue, consider the example in Figure 12 which shows three exercise vectors. The row dimension of the matrix depends on the number of exercise vectors, three in this example. Hence, the generation procedure should attempt to minimize the number of exercised vectors by generating vectors that exercise a large number of critical nets. Looking at the ATPG parallel, this is known as the Test Vector Compaction Problem [Bushnell and Agrawal 2000].

The column dimension contains the exercise data feeding each new MUX (up to 10 in this example). A straight-forward implementation requires a ROM of size 3×10 and 10 new MUXes for this example. However, we observe that each multiplexer's data can fall in one of three categories. In the first category, all data have the "don't care" value (columns 3 and 10 in Figure 12(a)). These columns can be removed from the ROM. Furthermore, no MUX is necessary for these signals. In the second category, we have columns that can assume either a constant '*0*' or a constant '*1*' value (columns 2, 4, 7, 8). These columns can also be removed from the ROM but still require a corresponding MUX set to the constant value. In the third category, both an MUX and an ROM column are needed, as the value of the MUX data varies among different vectors (columns 1, 5, 6, 9 in Figure 12(a)). We define all MUXes in the first category as MUX_X, those in the second category as MUX_0 + MUX_1, and finally those in the last category as MUX_{ROM}. Using the preceding analysis, the final ROM size in this example is (3×4). The number of necessary MUXes is the number of signals driven by an ROM column plus the number of columns with constant values computed by $MUX_{ROM} + MUX_1 + MUX_0$, which is $4 + 3 + 1 = 8$ ($MUX_{ROM} = \{l_1, l_5, l_6, l_9\}$, $MUX_1 = \{l_2, l_4, l_7\}$, $MUX_0 = \{l_8\}$, $MUX_X = \{l_3, l_{10}\}$). Clearly, the existence of "don't care" bits (X) in the vector set enables ROM compaction towards the column dimension as well as reduction of the necessary new MUXes.

Hence, the vector generation procedure should aim towards a compacted vector set to exercise the critical nets which, (a) has a small number of vectors and (b) has a large number of "don't care" bits in each vector. Such an approach is described next in Section 6.3.

6.3. Generation of Exercise Vectors with Large Number of Unspecified Bits

The proposed vector generation algorithm is outlined in Figure 13. As already stated, the overall goal is to generate a small number of vectors, each with a large number of unspecified bits, which exercise all nets on the critical path logic. The input to the algorithm is the critical path logic of the router $\boldsymbol{R}$ and the list of critical nets $\boldsymbol{N}$ with corresponding duty cycles $\boldsymbol{D}$. Priority is given to nets with high duty cycle, even

Procedure Exercise Vector Generation ( )
Inputs: Baseline router netlist $\boldsymbol{R}$, critical nets list $\boldsymbol{N}$, duty cycle per critical net $\boldsymbol{D}$
Outputs: Set of exercise vectors $\boldsymbol{V}$, list of exercised critical nets $\boldsymbol{N}_e$

```
01: Sort the elements of the critical nets list N based on D
02: N_e = NULL;                          // list of exercised critical nets
03: N_red = NULL;                        // list of redundant critical nets
04: j=1;                                 // exercise vector index
05: while (N ≠ ∅)
06:    v_j = X;                          // initialize v_j with all unassigned values ("don't cares")
07:    ∀ critical net n_i ∈ N            // for each net not exercised yet
08:       v_j'= justify (R, n_i, v_j);   // justify additional values of v_j in order to exercise n_i
09:       if (v_j' != NULL )
10:          add n_i in N_e and delete n_i from N
11:          simulate v_j' on R
12:          ∀ n_k ∈ N                  // for each net not exercised yet
13:             if (n_k == 1)
14:                add n_k in N_e and delete n_k from N
15:          v_j = v_j';                // update current vector
16:       else
17:          add n_i to N_red and delete n_i from N
18:    add v_j in V
19:    j++;
20: return V, N_e;
```

Fig. 13. Deterministic vector generation algorithm.

though all nets are considered. The output of the algorithm is a set of vectors $\boldsymbol{V}$ and a list of critical nets exercised $\boldsymbol{N_e}$. Starting with a vector with all unassigned values ($v_j = X$), the algorithm iteratively (lines 7–17) attempts to exercise as many critical nets as possible by justifying values on the current vector v_j. After each successful justification, the vector (v_j) is simulated to check for the existence of additional critical net activations that can also be exercised by v_j, which are then deleted from $\boldsymbol{N}$. When no more nets can be exercised, the generated vector v_j is added to the final vector set $\boldsymbol{V}$ and the procedure is repeated again (line 5) with a completely new vector (with all unassigned inputs) until all the critical nets are exercised ($\boldsymbol{N}$ is empty) or are classified as redundant (N_{red}). Redundant nets are the nets that cannot be exercised under any input assignment, and identification of those nets can indicate a possible problem in the synthesis of the router. We did not have any redundant nets in the extracted critical path logic circuit, but the proposed algorithm also covers this case for completeness purposes.

The proposed algorithm has two goals. First, it generates a small number of vectors. This is achieved because each vector is forced to exercise as many critical nets as possible by explicitly targeting them and furthermore simulating the vector values for any other critical nets that may be exercised without explicitly being targeted during each iteration (lines 7–17). The second goal is to have a large number of unspecified bits in the generated vectors in order to optimize the hardware overhead via compaction of exercise data using the techniques discussed in Section 6.2. This is achieved by using a variant of a powerful in-house PODEM-based ATPG justification procedure [Goel 1981; Neophytou and Michael 2010]. The justification procedure (line 8) is executed iteratively and specifies only the necessary vector bits during each iteration. In this manner, the generated vector contains a large number of "don't care" bits.

6.4. Vectors Generation Results and Underlying Exercise Logic

Figure 14 shows the additional exercise mode logic added to the extracted critical path logic of the baseline router. The extracted critical path logic circuit consists of 1,435 inputs, 357 outputs connected to flip-flops inside VCs as shown in Figure 9, and 14,653

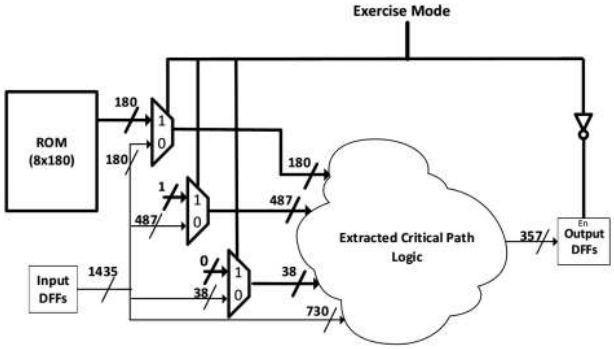


Fig. 14. Critical path logic with exercise logic (shown with darker lines), after vector generation.

internal nodes. From the extracted circuit, the critical path logic consists of 732 critical nets which need to be exercised. Using the deterministic vector generation algorithm proposed in Section 6.3, eight vectors are generated which exercise all of the 732 critical nets at least one time (some of them are exercised more than once). After the generation of the vectors, we follow a similar procedure to that discussed in Section 6.2 in order to optimize the hardware overhead (ROM size and number of MUXes). From 1,435 inputs which correspond to possible MUX locations, 730 have "don't care" values (see MUX_X in Section 6.2) and can be removed from the ROM, while 38 can be set to constant value *'0'* (MUX_0) and 487 can be set to constant value *'1'* (MUX_1). Therefore, the necessary ROM size is (8×180) $(= 1{,}435 - 730 - 38 - 487)$ with 705 $(= 180 + 38 + 487)$ MUXes (525 of the MUXes are having a constant value on their input pin) shown in Figure 14.

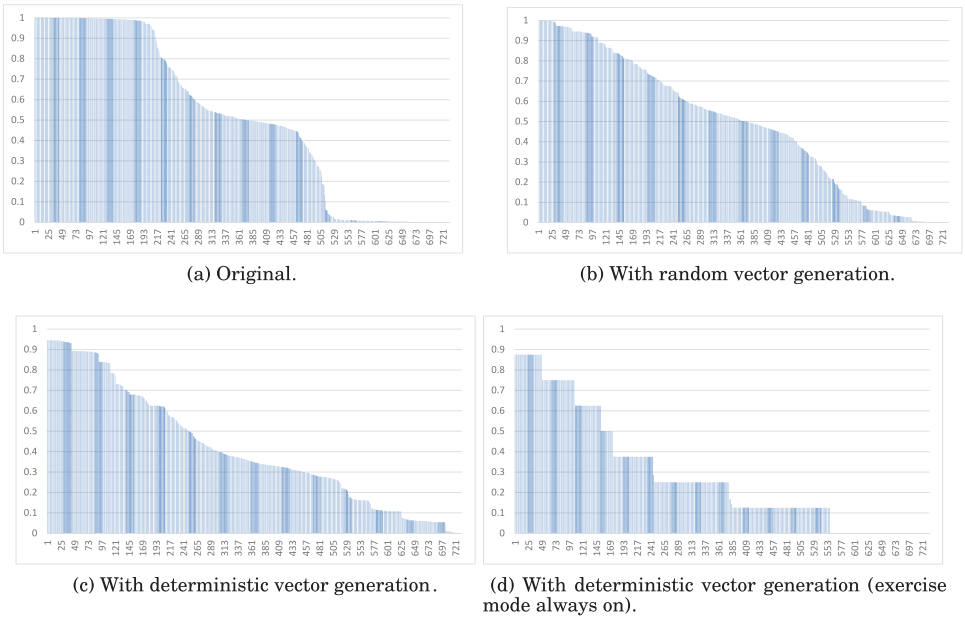
7. EVALUATION

In this section, we first outline our experimental setup. This is followed by a detailed exploration of the benefits and costs of our proposed technique.

7.1. Experimental Setup

The baseline router, adapted from RTL code made publicly available by Becker [2012] contains three pipeline stages. The detailed parameters of the router are listed in Table I. It is synthesized using the Synopsys Design Compiler mapped to a 45nm technology library at 1GHz. We note that the added exercise circuit as mentioned in Section 5 is largely off the critical path. Thus, router synthesis produces the same clock frequency as the baseline router of 1GHz. The critical paths were extracted using Synopsys Design Vision. All paths with $\leq$10% slack were retained and analyzed. The wire activity along the paths is extracted with Synopsys' Verilog Compiler Simulator (VCS) and analyzed offline. The power consumption is evaluated using PrimeTime.

The router is evaluated under both synthetic and realistic workloads. The realistic workloads are captured as traces from the gem5 simulator [Binkert et al. 2011] emulating a 64-core system executing multithreaded programs from the PARSEC v2.1 suite. Table I summarizes the system configuration. We compute the incoming rate of each router over the entire application execution, in an 8×8 mesh network, individually under X-Y DOR routing. The per-router min, max, and average incoming rates for each application were calculated. Random traffic is generated at these incoming rates and is applied to the synthesized router to extract the activity of its wires. This methodology gives an estimate for realistic workloads, such as those of the PARSEC benchmark suite. For both synthetic and realistic workloads, we execute the post-synthesis models of both the baseline and proposed routers, for 100,000 cycles, to measure the wire activity.



(a) Original.

(b) With random vector generation.

(c) With deterministic vector generation.

(d) With deterministic vector generation (exercise mode always on).

Fig. 15. Duty cycles of critical path nodes with 2% incoming flit rate, sorted from highest to lowest.

7.2. Experimental Results

Random versus Deterministic Vector Generation. Aging due to NBTI depends on the duty cycles of nodes along the critical paths. We study the impact of randomly generated vector sets to exercise the critical path nodes. Here we used a set 16 of 1,435-bit random vectors to drive the exercise logic. Sixteen vectors were used as more random vectors did not appear to provide any further reduction in duty cycle. Figure 15 shows the duty cycles of the nodes on critical paths under different scenarios. Here, all simulations are performed under synthetic traffic of 0.02 flits/cycle. As Figure 15(a) shows, the duty cycles for a baseline router are biased towards either "1" or "0." The nodes with duty cycle close to 1 significantly affect the aging due to NBTI, as shown in Equation (11). Figure 15(b) shows that using random vectors to exercise the critical paths produces improvement, but there are still a number of nodes with duty cycle of $\sim$1. We note that here, we must have an exercise vector which is of the same bit-width as the number of inputs to the critical path logic (1,435 bits), hence requiring 1,435 random bits per vector in the ROM.

As Figure 15(d) shows, the duty cycles improve greatly when the vectors used during exercise mode are generated using the deterministic method described in Section 6. After optimization, just eight vectors—each 180-bit wide—are enough to exercise all the nodes at least once. The ROM size of 8×180 will also be much smaller when compared to that of $16 \times 1{,}435$ for randomly-generated vectors. When the exercise mode is always on, the maximum duty cycle that a node can have is 0.875 (7/8), which confirms that all the nodes are exercised at least by one of the generated vectors. In Figure 15(c), when synthetic traffic of 0.02 flits/cycle is added to the generated vectors, none of the nodes have a duty cycle of 1, though the results are smoothed somewhat from Figure 15(d).

Aging under Synthetic Workloads. We now examine the potential gain in router lifetime of the proposed technique versus a baseline for a range of arbitrary incoming rates. As previously discussed, the per-router incoming rate under PARSEC workloads varies between 0.0005 (*x264*) to 0.085 (`canneal`). Figure 16 shows the normalized acceleration factor (Equation (13)) versus the baseline router at the same incoming rate.

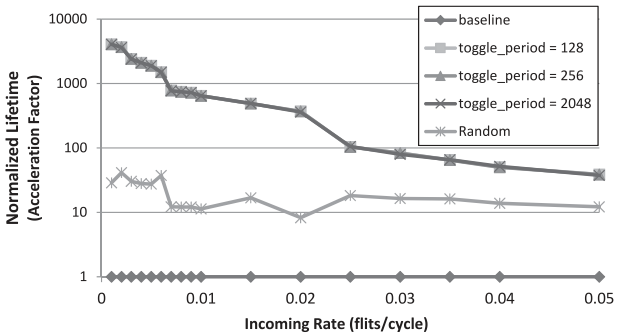


Fig. 16. Normalized lifetime (acceleration factor) for router under a given synthetic incoming rate ranging from 0.001 flits/cycle to 0.05 flits/cycle.

As explained in Section 4, the acceleration factor gives the lifetime of the system under consideration, normalized to the lifetime of the reference system. In Figure 16, the term "baseline" refers to the lifetime of the baseline router, normalized to 1, while the term "Random" indicates that the vectors in 16-entry ROM are randomly generated, while the rest of them indicate cases with deterministic vector generation with different vector rotation periods. "*Toggle_period* = *X*" indicates use of the generated vectors with a rotation period of *X* cycles from one vector to the next. The normalized lifetime is plotted on a logarithmic scale.

Lifetime improves dramatically for the routers with low incoming rates, as Figure 16 shows. Generally, low incoming rates cause a greater bias in the duty cycle and hence allow more room for the improvement; thus the greatest gains in lifetime occur with the lowest incoming rates. It is quite evident that the deterministic generation of vectors gives significantly higher improvement in lifetime when compared to random generation. The lifetime improvement with random vectors is not a monotonically decreasing curve as in the case of deterministic vectors. This is because we consider the worst case path of all paths within 10% slack in our calculations. When random vectors are used to exercise the paths, the worst-case path differs for each individual incoming packet rate. This will not happen if deterministic vectors are used, because an individual node will have the set of values under exercise mode.

Figure 16 shows no significant difference in lifetime between the three different vector rotation periods. This is because the duty cycle for a particular node on a critical path will remain the same if the same set of vectors are repeated any number of times. It has to be noted that the exercise vector is changed only when it is in the exercise mode for a certain time indicated by a rotation period. For our simulation of 100,000 cycles and an incoming rate of 0.05 flits/cycle, a rotation period of 2,048 is the maximum that we can have so that each vector is used at least once. Hence, we use a rotation period of 2,048 for the remainder of this article, as this design point implies the lowest overhead in terms of activity factor.

Lifetime under PARSEC Workloads. Figure 17 depicts the normalized lifetime of the network using the proposed technique under the PARSEC workloads. The lifetime of the network is estimated by computing the acceleration factor of the router with the minimum incoming rate in the network, as it is the most susceptible to aging effects. The reference system is the baseline router receiving the same incoming rate.

Deterministic vector generation achieves an average of $\sim$2300$\times$ reduction in wear rate (bars marked "AVG") as compared to that of random vector generation which only gives $\sim$28$\times$ improvement. As expected, the proposed technique performs better when incoming rate is low. Figure 17 shows "ferret" and "x264" are the applications with the two lowest incoming rates in the PARSEC suite. Even when the average incoming rate is as high as 0.05 flits per cycle (`canneal`), the deterministic vector generation still achieves the normalized lifetime of 800$\times$ due to the extreme spread in per-router

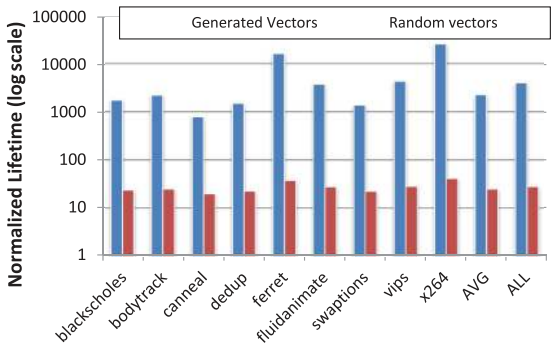


Fig. 17. Normalized lifetime of the network using the proposed technique under realistic workload.

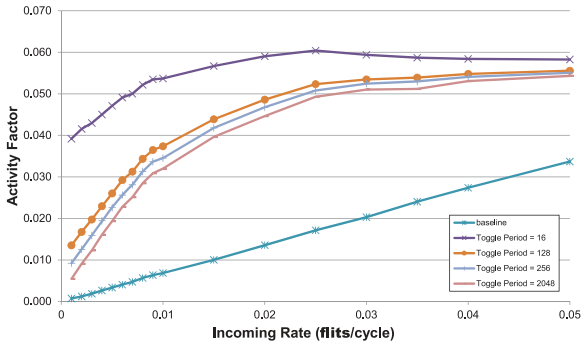


Fig. 18. Activity factor versus injection rate.

incoming rates from minimal to maximum seen in that application. The random vector generation does give a little improvement in lifetime, but it is no where close to what we can achieve with deterministic vector generation. The bars designated as "ALL" denote a case in which the system executes each of the applications sequentially one at a time. In this case, the improvement becomes $\sim 4{,}000\times$. We find that the execution times of "ferret" and "x264" are the longest among the applications, and hence the incoming rate for "ALL" is dominated by those applications.

Activity Factor. One potential downside of a technique that decreases the duty cycle along the critical path is that it could increase the activity factor as well, resulting in potential HCI-induced aging problems. Figure 18 shows the average activity factor along the critical paths at various flit incoming rates for different router models. For the "baseline" router, the activity factor is linearly proportional to the incoming rate, as the incoming flits are the only stimuli to the allocator. In the modified routers, the activity factor also increases as the incoming rate grows, but it increases slightly more rapidly than the "baseline" case. The growth of activity factor with respect to the incoming rate is more rapid at low incoming rates, as the exercise logic has more opportunities to become active. As the incoming rate increases and the exercise logic misses opportunities to generate a new random vector, the increase in the activity factor slows down.

Generally, there is a significant difference in activity factors between baseline and modified routers, even at high incoming rates. Each time exercise mode is turned on, many of the critical path nodes in a router switch to a different logic state, leading to a burst in activity. In Figure 18, the impact of rotation period on activity factor can be clearly observed. The increase in activity factor decreases with higher rotation periods. At incoming rates of 0.05 flits/cycle and above, the activity factor of the modified

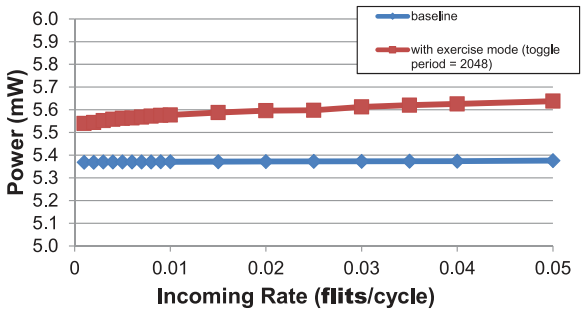


Fig. 19. Router power consumption versus injection rate. (*Note*: the Y-axis is scaled to provide detail.)

router with toggle period of 2,048 reaches a saturation point of ∼5%, implying that the proposed technique should not significantly impact HCI.

Power Analysis. An increase in activity factor in the allocators should be expected to lead to additional dynamic power consumption for the router. Further, the additional "exercise logic" should also require additional static and dynamic power. Thus we performed a power analysis of the baseline and proposed router designs using Synopsys PrimeTime. Figure 19 shows the power consumption with respect to varying incoming rates for the different router models. As expected, router power consumption increases as the incoming rate increases; however, we find that the router with exercise mode applied increases the total router power by less than 5% across all incoming rates. In part, this is because the major contribution for the power consumption in both the baseline router and the router with exercise mode applied is from sequential circuit elements (∼90%). During the exercise mode, only combinational circuit elements are switched, limiting the potential for power to increase. Also the additional exercise mode logic increases router area by less than 5%.

8. RELATED WORK

Aging models for transistors have been extensively studied since technology crossed the submicron border, which inherently made the CMOS manufacturing process vulnerable to runtime faults. NBTI and HCI are dominant wearout effects and have thus been more intensively studied [Oboril and Tahoori 2012]. Conventionally, aging effects are studied under DC stress conditions where it is easier to measure transistor parameters [JEDEC 2011; Li et al. 2008b; Maricau and Gielen 2009]. However, AC stress conditions are more realistic for high-frequency, long-term CMOS operation; hence some works [Karpuzcu et al. 2009; Tudor et al. 2011; Oboril and Tahoori 2012; Saluja et al. 2008] discuss such long-term models, while other works introduce the relationship between DC and AC stress conditions [Wang et al. 2011].

The newly emerging devices, such as multigate field effect transistors MuFETs and FinFETs [Auth 2012; Saitoh et al. 2012] at 22nm process technology and below, have gradually become an object for aging effects exploration. Wang et al. [2011] make an attempt towards presenting a unified aging model for the effects of both HCI and NBTI. They derive models for double- and triple-gate FinFETs for each of these aging effects that capture specific FinFET geometry aspects. Mahmoud et al. [2014] presents a comprehensive framework that assists in designing fortified VLSI gates at 16nm against BTI-induced aging degradation, while achieving energy savings.

Unfortunately, the vast majority of the reported models lack important details, such as values for various constants, measurement conditions, detailed explanation of parameters, etc. Thus, it becomes fairly challenging to employ the existing frameworks in the context of microarchitecture and to perform meaningful aging effect calculations. The recent work by Kleeberger et al. [2014] and Fang and Sapatnekar [2014] do propose combined models for estimating the additional aging-induced delay in combinatorial

circuits; however, the parameters they obtained are tuned to a particular technology node different from the one used here and hence cannot be generalized.

Various techniques have been proposed to mitigate the aging effect in processor core architectures. Among those proposed mechanisms, Gunadi et al. [2010] suggested the Colt duty cycle equalizer which balances the duty cycle by alternating true and one's complement data representations. Abella et al. [2007] introduce the Penelope NBTI-aware processor architecture where they discuss a number of techniques for combating NBTI, including a mechanism which writes special values in memory cells in order to keep the duty cycle at an ideal 50%. Gupta and Sapatnekar [2013] propose generating idle periods for BTI recovery by power gating most of the components in a single-core processor system. These idle periods are generated by running the core at higher-than-nominal frequency. Guo et al. [2014] propose an approach for accelerating NBTI recovery by applying a negative supply voltage during idle periods of a chip. Kumar et al. [2006] proposed periodic cache flipping so as to provide periods of relaxation for the influenced pMOSFET, allowing dynamic recovery of the threshold voltage level. Oboril and Tahoori [2014] proposed reducing aging in microprocessor pipelines by replacing the traditional design-time time-balancing scheme of pipelines with MTTF-balanced pipelines also at design time, hence achieving targeted MTTF values; this technique also allows for higher operational frequencies at reduced energy expenditures. The same authors target both HCI and BTI effects; however, it is unclear how they balance between them. Next, Lai et al. [2014] analyzed the effects of BTI on the clock distribution network in a microprocessor with clock gating features and then proposed two BTI-Gater cells, similar in function to the exercising mode multiplexers used in our work, to balance delay degradation on the gated clock branch. Unlike in our work, their technique requires a software-based sleep scheduling wrapper that works in conjunction with the BTI-gater cells to reduce aging, an overhead that our work excludes as it is based on periodic use of deterministically-derived exercise vectors to achieve NoC aging reduction in multicores. Bild et al. [2012] proposed the Internal Node Control (INC) scheme to reduce the impact of static NBTI on circuits with frequently idle functional units such as adders, subtractors, and shifters. INC placements allow outputs of an INC-modified gate to be forced to specific values during sleep mode, and as in our case, exercise various paths to combat NBTI.

The aforementioned three works bear similarities to the "exercise logic" proposed in this article, in the context that these techniques periodically insert inverted or random values to balance the duty cycle. None of the three, however, deals with NoC router microarchitectures which are critical to the system's survivability.

Although our approach is similar to the approaches in the aforementioned works, here we actually handle a different problem. In previous works, the researchers focused on the duty-cycle bias of the data paths, while we mainly balance the uneven duty cycles along the control paths. In fact, the sources of the uneven duty cycles are different. In previous works, they are dominant in biased data contents, while in our work, it is more evident during the proved extremely low activity seen in NoC routers (see Section 3). Although the proposed "exercise logic" is also able to resolve the aging problems due to the skewed data content along the data path of the NoC routers, we utilize it only for the timing-critical paths where the NBTI impact occurs in its most critical form.

A number of works attempt to develop a reliability model at the architectural level. Srinivasan et al. [2004] proposed such a model of a processor core, which considers a set of failure mechanisms. Their assumptions, such as even distribution of failures across failure mechanisms and uniform failure rate across the design, restrict the accuracy of the model when extended to the entire chip. Shin et al. [2007] further develop this concept. They introduce *effective defect density* and *effective stress condition* coefficients that weigh the failure impact across the chip area and runtime, respectively. Shin et al. illustrated their approach under several failure mechanisms and presented indicative weight coefficients for a set of abstract architectural structures. By contrast, in this work, we examine the actual critical path of a particularly failure-sensitive, vital CMP component—the NoC router. We show that under realistic workloads, this component

is highly susceptible to NBTI-based wear, and we develop a technique to mitigate this wear.

Aging has been also examined in the NoC domain. Bhardwaj et al. [2012, 2013] propose routing algorithms to mitigate multiple aging mechanisms. They also point out that NBTI plays a major role in NoC router aging, and their routing techniques balance the traffic load across the network to level out the aging rates among the routers. Their approach is reasonable in that they force the network traffic to detour through routers of low utilization which, on the contrary, accelerates NBTI-caused aging. However, they use these routing techniques for the opposite effect; the routing algorithms are actually designed to reduce the workload onto the routers which exhibit high utilization, which as we show here are not actually the routers likely to exhibit the most stress-related aging. Fu et al. [2010] propose a similar technique to ours in that it inserts special values to idle arbiters to mitigate NBTI. However, they propose this technique to make arbiters less frequently utilized so as to give these routers a chance to recover from the effects of NBTI, which is actually not necessarily applicable to frequently utilized circuits.

In these previous NoC-oriented studies, it is assumed that the NBTI stress time is proportional to the router utilization; however, on the contrary, we prove that this is not the actual case. Through detailed, gate-level analysis not found in earlier works, we demonstrate that the duty cycle becomes more skewed when the NoC router is actually underutilized and not when it is highly- or overutilized.

9. CONCLUSIONS

The NoC interconnect is critical to the survival of the CMP system. In this article, we develope a critical path model for NBTI-induced wearout after analyzing HCI- and NBTI-induced wear due to stresses caused by realistic workloads and apply them onto the interconnect microarchitecture. A key finding from this modeling is that, counter to prevailing wisdom, wearout in the CMP on-chip interconnect is correlated with lack of load observed in the NoC routers, rather than high load. We then develope a novel wearout-decelerating scheme in which routers under low load have their wearout-sensitive components exercised without significantly impacting cycle time, pipeline depth, power consumption, or area of the overall router. The exercise mode data are generated deterministically for maximum impact. We subsequently show that the proposed design yields a $\sim$2,300$\times$ decrease in router wear due to NBTI. In our future work, we plan to further explore degradation due to electromigration in the routers and links between the routers.

REFERENCES

Jaume Abella, Xavier Vera, and Antonio González. 2007. Penelope: The NBTI-aware processor. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 85–96.

M. Abramovici, M. A. Breuer, and Arthur D. Friedman. 1990. *Digital Systems Testing and Testable Design*. AT&T Bell Laboratories and W.H. Freeman.

M. Agarwal, B. C. Paul, Ming Zhang, and S Mitra. 2007. Circuit failure prediction and its application to transistor aging. In *Proceedings of the 25th IEEE VLSI Test Symposium*. 277–286.

Chris Auth. 2012. 22-nm fully-depleted tri-gate CMOS transistors. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*. 1–6.

Daniel U. Becker. 2012. Efficient microarchitecture for network-on-chip routers. Ph.D. Dissertation, Stanford University.

K. Bhardwaj, K. Chakraborty, and S. Roy. 2012. An MILP-based aging-aware routing algorithm for NoCs. In *Proceedings of the Design, Automation Test in Europe Conference (DATE)*. 326–331.

K. Bhardwaj, K. Chakraborty, and S. Roy. 2013. Towards graceful aging degradation in NoCs through an adaptive routing algorithm. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 382–391.

Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 72–81.

David R. Bild, Robert P. Dick, and Gregory E. Bok. 2012. Static NBTI reduction using internal node control. *ACM Trans. Des. Autom. Electron. Syst.* 17, 4 (2012).

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. *ACM SIGARCH Comput. Archit. News* 39, 2 (2011).

Jason Blome, Shuguang Feng, Shantanu Gupta, and Scott Mahlke. 2007. Self-calibrating online wearout detection. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 109–122.

Michael Bushnell and Vishwani D. Agrawal. 2000. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Frontiers in Electronic Testing, Vol. 17. Springer.

William J. Dally and Brian Towles. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 684–689.

Andrew DeOrio, Kostantinos Aisopos, Valeria Bertacco, and Li-Shiuan Peh. 2011. DRAIN: Distributed recovery architecture for inaccessible nodes in multi-core chips. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*.

Jianxin Fang and Sachin S. Sapatnekar. 2014. Incorporating hot carrier injection effects into timing analysis for large circuits. *IEEE Trans. VLSI Syst.* 22, 12 (2014), 2738–2751.

David Fick, Andrew DeOrio, Gregory Chen, Valeria Bertacco, Dennis Sylvester, and David Blaauw. 2009. A highly resilient routing algorithm for fault-tolerant NoCs. In *Proceedings of the Design, Automation Test in Europe Conference (DATE)*.

Xin Fu, Tao Li, and José A. B. Fortes. 2010. Architecting reliable multi-core network-on-chip for small scale processing technology. In *Proceedings of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

Prabhakar Goel. 1981. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Trans. Computers* 100, 3 (1981), 215–222.

Erika Gunadi, Abhisek A. Sinkar, Nam Sung Kim, and Mikko H. Lipasti. 2010. Combating aging with the colt duty cycle equalizer. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 103–114.

Xinfei Guo, Wayne Burleson, and Mircea Stan. 2014. Modeling and experimental demonstration of accelerated self-healing techniques. In *Proceedings of the 51st Annual Design Automation Conference (DAC'14)*. 1–6.

Saket Gupta and Sachin S. Sapatnekar. 2013. Employing circadian rhythms to enhance power and reliability. *ACM Trans. Des. Autom. Electron. Syst.* 18, 3 (2013).

J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. K. De, and R. V. Der Wijngaart. 2011. A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. *IEEE J. Solid-State Circuits* 46, 1 (2011).

Lin Huang and Qiang Xu. 2010. Agesim: A simulation framework for evaluating the lifetime reliability of processor-based socs. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. 51–56.

ITRS. 2009. Process integration, devices, and structures (PIDS). International Technology Roadmap for Semiconductors.

JEDEC. 2011. Failure mechanisms and models for semiconductor devices, JEP122G. http://www.jedec.org/.

U. R. Karpuzcu, B. Greskamp, and J. Torrellas. 2009. The BubbleWrap many-core: Popping cores for sequential acceleration. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 447–458.

Veit B. Kleeberger, Martin Barke, Christoph Werner, Doris Schmitt-Landsiedel, and Ulf Schlichtmann. 2014. A compact model for NBTI degradation and recovery under use-profile variations and its application to aging analysis of digital integrated circuits. *Microelectron. Reliab.* 54, 6 (2014), 1083–1089.

Kelin J. Kuhn. 2007. Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale CMOS. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*. 471–474.

Sanjay V. Kumar, Chris H. Kim, and Sachin S. Sapatnekar. 2006. Impact of NBTI on SRAM read stability and design for reliability. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED)*.

Liangzhen Lai, Vikas Chandra, Robert Aitken, and Puneet Gupta. 2014. BTI-Gater: An aging-resilient clock gating methodology. *IEEE J. Emerg. Select. Topics Circ. Syst.* 4, 2 (2014).

Xiaojun Li, Jin Qin, and J. B. Bernstein. 2008b. Compact modeling of MOSFET wearout mechanisms for circuit-reliability simulation. *IEEE Trans. Device Mater. Reliab.* 8, 1 (2008), 98–121.

Yanjing Li, Samy Makar, and Subhasish Mitra. 2008a. CASP: Concurrent autonomous chip self-test using stored test patterns. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. 885–890.

Yinghai Lu, Li Shang, Hai Zhou, Hengliang Zhu, Fan Yang, and Xuan Zeng. 2009. Statistical reliability analysis under process variation and aging effects. In *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC)*. 514–519.

Mohamed Mounir Mahmoud, Norhayati Soin, and Hossam A. H. Fahmy. 2014. Design framework to overcome aging degradation of the 16 nm VLSI technology circuits. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 33, 5 (2014).

E. Maricau and G. Gielen. 2009. A methodology for measuring transistor ageing effects towards accurate reliability simulation. In *Proceedings of the 15th IEEE International On-Line Testing Symposium (IOLTS)*. 21–26.

S. Nassif, K. Bernstein, D. J. Frank, A. Gattiker, W. Haensch, B. L. Ji, E. Nowak, D. Pearson, and N. J. Rohrer. 2007. High performance CMOS variability in the 65nm regime and beyond. In *Proceedings of the IEEE the International Electron Devices Meeting (IEDM)*. 569–571.

Stelios N. Neophytou and Maria K. Michael. 2010. Test set generation with a large number of unspecified bits using static and dynamic techniques. *IEEE Trans. Comput.* 59, 3 (2010), 301–316.

F. Oboril and M. B. Tahoori. 2012. ExtraTime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level. In *Proccedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 1–12.

Fabian Oboril and Mehdi B. Tahoori. 2014. Aging-aware design of microprocessor instruction pipelines. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 33, 5 (2014).

Li-Shiuan Peh and William J. Dally. 2001. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*. 255–266.

Michael D. Powell, Arijit Biswas, Shantanu Gupta, and Shubhendu S. Mukherjee. 2009. Architectural core salvaging in a multi-core processor for hard-error tolerance. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*. ACM.

M. Saitoh, K. Ota, C. Tanaka, Y. Nakabayashi, K. Uchida, and T. Numata. 2012. Performance, variability and reliability of silicon tri-gate nanowire MOSFETs. In *Proceedings of the IEEE International Reliability Physics Symposium (IRPS)*. 6A–3.

T. Sakurai and A. R. Newton. 1990. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE J. Solid-State Circ.* 25, 2, 584–594.

K. K. Saluja, S. Vijayakumar, W. Sootkaneung, and X. Yang. 2008. NBTI degradation: A problem or a scare? In *Proceedings of the International Conference on VLSI Design*. 137–142.

Timo Schonwald, Jochen Zimmermann, Oliver Bringmann, and Wolfgang Rosenstiel. 2007. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD)*. 527–534.

J. Shin, V. Zyuban, Zhigang Hu, J. A. Rivers, and P. Bose. 2007. A framework for architecture-level lifetime reliability modeling. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 534–543.

Michael A. Skitsas, Chrysostomos A. Nicopoulos, and Maria K. Michael. 2013. DaemonGuard: O/S-assisted selective software-based self-testing for multi-core systems. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 45–51.

Jared C. Smolens, Brian T. Gold, James C. Hoe, Babak Falsafi, and Ken Mai. 2007. Detecting emerging wearout faults. In *Proceedings of the IEEE Workshop on Silicon Errors in Logic-System Effects (SELSE)*.

J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. 2004. The case for lifetime reliability-aware microprocessors. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*. 276–287.

B. Tudor, J. Wang, Zhaoping Chen, R. Tan, Weidong Liu, and F. Lee. 2011. An accurate and scalable MOSFET aging model for circuit simulation. In *Proceedings of the 12th International Symposium on Quality Electronic Design (ISQED)*. 1–4.

Yao Wang, S. Cotofana, and Liang Fang. 2011. A unified aging model of NBTI and HCI degradation towards lifetime reliability management for nanoscale MOSFET circuits. In *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. 175–180.

Zhen Zhang, Alain Greiner, and Sami Taktak. 2008. A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip. In *Proceedings of the 45th ACM/IEEE Design Automation Conference (DAC)*. 441–446.

Received July 2014; revised December 2014, March 2015; accepted May 2015