USE IT OR LOSE IT: PROACTIVE, DETERMINISTIC LONGEVITY IN FUTURE CHIP MULTIPROCESSORS

A Thesis

by

SIVA BHANU KRISHNA BOGA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2015

Major Subject: Computer Engineering

ABSTRACT

Ever since the VLSI process technology crossed the sub-micron threshold, there is an increased interest in design of fault-tolerant systems to mitigate the wearout of transistors. Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) are two prominent usage based transistor degradation mechanisms in the deep sub-micron process technologies. This wearout of transistors can lead to timing violations along the critical paths which will eventually lead to permanent failures of the chip. While there have been many studies which concentrate on decreasing the wearout in a single core, the failure of an individual core need not be catastrophic in the context of Chip Multi-Processors (CMPs). However, a failure in the interconnect in these CMPs can lead to the failure of entire chip as it could lead to protocol-level deadlocks, or even partition away vital components such as the memory controller or other critical I/O. Analysis of HCI and NBTI stresses caused by real workloads on interconnect microachitecture shows that wearout in the CMP on-chip interconnect is correlated with lack of load observed in the network-on-chip routers. It is proven that exercising the wearout-sensitive components of routers under low load with random inputs can decelerate the NBTI wearout. In this work, we propose a novel deterministic approach for the generation of appropriate exercise mode data to maximize the life-time improvement, ensuring design parameter targets are met. The results from this new proposed design yields $\sim$2300$\times$ decrease in the rate of CMP wear due to NBTI compared to that of $\sim$28$\times$ decrease shown by previous work.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Paul Gratz, for the support and guidance during the course of my study here at Texas A&M. I am very thankful to him for instilling positive thinking and patience in me which are much needed in research.

Secondly, I would l ike t o thank Dr. Maria K. Michael, Stavros Hadjitheo-phanous from University of Cyprus, and Dr. Vassos Soteriou from Cyprus University of Technology for their valuable input and assistance in finishing my work on time.

I would like to thank my committee members, Dr. Peng Li and Dr. Duncan M. "Hank" Walker, for their interest in my work. Their insightful comments have helped improve the quality of my work.

I would like to thank my mentor at Intel during my internship , Tom Skrzeszewski, in helping me improve my analytical and presentation skills. I would also like to thank him for being flexible enough to attend my research meetings during the work hours.

I am very fortunate to be part of CAMSIN research group. The diverse projects in the group expanded my breadth of knowledge in the field of computer architecture.

Last but not least, I would like to thank my parents for their unwavering support in letting me pursue my goals. I am forever indebted to them for giving me the freedom to make my own decisions and their trust in my choices.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Moore's law scaling still continues and the size of transistor is getting smaller with every process generation. The number of transistors that can be integrated on a single chip today is greater than ever before. Ever since hitting the power wall, the design trend has been to utilize the increasing supply of transistors to develop multi-core processors. These multi-core processors contain either homogeneous or heterogeneous processing cores which are connected by an on-chip interconnect which is typically organized as a Network-On-Chip(NoC) [12]. Figure 1.1 shows a Chip-MultiProcessor (CMP) design by Intel which has 48 processing cores [22].

As the transistors get smaller, they also become less reliable. There are several usage based transistor degradation mechanisms which wears out the transistors and hence limit the useful life-span of the Chip-MultiProcessors. A recent study by ITRS indicates that a 10-fold decrease of transistor wear-rate will be needed in the next 10 years to maintain current design lifetimes [16]. Unfortunately, no manufacturable solutions are known to handle this problem. Hence, fault-tolerant designs are needed at upper levels of design hierarchy to tackle this problem.

Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) are two most important usage based transistor degradation mechanisms. Both HCI and NBTI causes a shift in threshold voltage of the transistors [3] which are under stress and hence makes the transistor switching slower. This slow-down of transistors may result in timing violations along the critical paths and hence may render the entire chip useless. The modern chip designs account for this slow-down by adding a guard band (typically 10%) to the clock-period. The useful lifetime of a chip is defined as the time by which the transistors along the critical path wear-out to cause

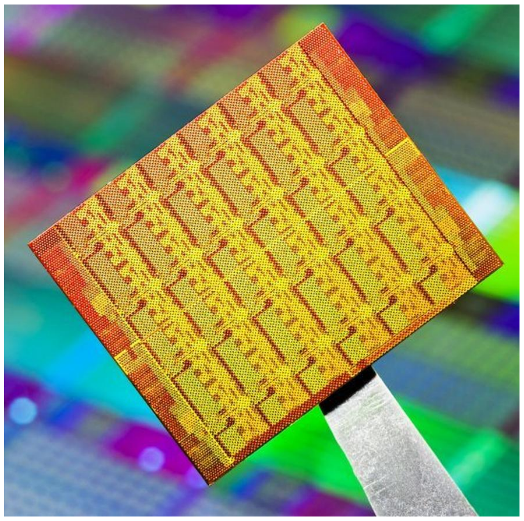


Figure 1.1: Intel's 48 core single-chip cloud computer.

timing violations even beyond this guard band.

Figure 1.2 shows a 64-core chip-multiprocessor with various peripherals connected to it. The homogeneous processing cores are connected by a Network-on-chip organized as a 2D mesh. Figure 1.2 [25] also shows different failure scenarios which may happen on such a system. Because of inherent core redundancy, an individual core failure need not be catastrophic to the functionality of the CMP provided one can do a fault-tolerant task migration to other processing cores [10, 41, 38, 30, 23]. This only results in lesser system throughput compared to the original system. However, as illustrated by next three scenarios, a failure in interconnect can be catastrophic to the CMP functionality. The next scenario shows a case where a particular link failure may disconnect a peripheral from the system. The next scenario shows an even worse case where a set of failures in the interconnect disconnects an entire portion of chip. A single link failure may also lead to routing deadlocks when the routing algorithm used is not adaptive to link failures. Hence even a single failure in the interconnect may render the entire chip useless.

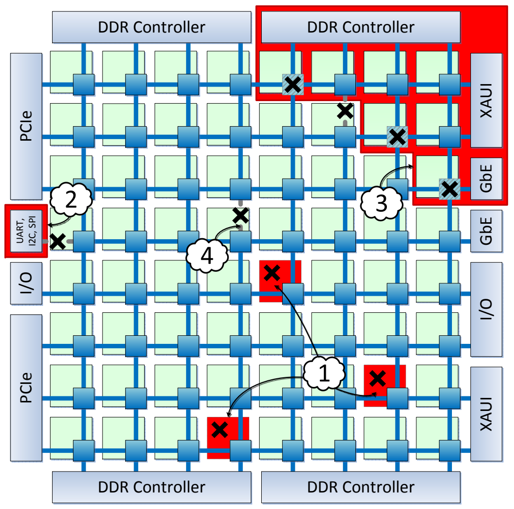


Figure 1.2: A 64-core CMP interconnected with an 8×8 2D mesh NoC illustrating different failure scenarios. (H.Kim *et al.* [25])

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 [44, 40, 15, 6, 5, 13], however, network isolation and key resource partitioning 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 the reactive mechanisms, in the hope that the latter would be required less frequently as faults in the system would occur less frequently.

The work done by H.Kim *et al.* [25] present one such proactive technique, designed to decelerate the effects of aging in the NoC of a CMP. To combat the aging effects caused by HCI and NBTI in current and future CMP on-die interconnect, they have characterized the dependence of application workload observed in NoC routers which are directly dependent upon HCI- and NBTI-induced wearout, and developed microarchitectural techniques to address the stresses that lead to these wearout mechanisms, in an attempt to significantly prolong the functional lifetime of the entire multicore system.

Based upon detailed HCI and NBTI transistor-level aging models, they developed a novel, critical path-based model to characterize the effects of aging-related wear. Based upon this model, the NoC router microarchitecture was analyzed to find the paths most susceptible to wearout. Using real workloads from the PARSEC benchmark suite [7], various wearout mechanisms that map onto those paths were characterized. This characterization shows that the major wearout mechanism in the NoC router circuits is NBTI because of biased duty cycles along these paths. The same work proposed a novel wearout-resistant router micro-architecture, which prolongs circuit lifetime through targeted mitigation techniques to reduce NBTI wearout. The NBTI wearout is reduced by exercising the wearout sensitive pipeline stage during the idle periods of operation using random inputs. The results from this work show an average improvement of $\sim 28\times$ compared to the baseline router.

The current work presented in this thesis aims at improving the technique presented by H.Kim *et al.* [25] which is mentioned above. As part of this work we propose 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. The results show an average relative lifetime improvement of $\sim 2300\times$ which is orders of magnitude better than the results of previous work.

This rest of the thesis is organized as follows. Chapter 2 provides a brief description of trasistor-level models for HCI and NBTI induced wear which were used by the H.Kim *et al.* and gives a brief summary of the workload characterization study done by H.Kim *et al.* [25]. Chapter 3 proposes a novel deterministic way to maximize the lifetime improvement achieved by prior work,which utilizes the exercise mode data derived by the systematic methodology described in chapter 4, while chapter 5 evaluates the proposed design. Finally chapter 6 presents prior related work, while chapter 7 concludes this work.

2. BACKGROUND[*]

As already mentioned in Chapter 1, prior work done by H.Kim *et al.* [25] has proposed a lifetime extending router micro-architecture for future Chip-MultiProcessors. In this chapter, we will present a brief description on the transistor degradation models which were used by this prior work. Then we will summarize the work load characterization of router circuits which are typically used in interconnects of CMPs done by H.Kim *et al.* [25].

2.1 Failure mechanisms

Two major usage based transistor degradation mechanisms in sub-micron process technologies are Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI). Both these mechanisms slows down the transistor switching speed which lead to timing violations along critical paths of the circuit. This section provides a brief overview of these transistor degradation mechanisms.

2.1.1 Hot carrier injection

Hot Carrier Injection (HCI) is an usage based transistor degradation mechanism which affects both PMOS and NMOS transistors. When current flows through the channel, the charge carriers gain sufficient kinetic energy to embed in gate oxide. This results in trap generation and leads to a shift in transistor parameters [3].

Since current flow happens only during the transition of voltage level from high to low or low to high, the time under HCI stress is proportional to the activity factor on the device. Based on the Reaction-Diffusion model which uses the threshold voltage

[*]Part of this chapter is reprinted with permission from *Use it or lose it: wear-out and lifetime in future chip multiprocessors* by Kim, Hyungjun and Vitkovskiy, Arseniy and Gratz, Paul V and Soteriou, Vassos, in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, 2013*, pages 136-147, Copyright 2013 by ACM

(V_{th}) shift as a proxy of HCI stress [43], H.Kim *et al.* [25] have derived the equation for lifetime for a single transistor under AC stress as shown below.

$$TTF_{HCI}(T,\alpha_{SA})\big|_{AC} = A_{HCI}\frac{1}{d_g f \alpha_{SA}}\left(I_{sub}\right)^{-N'} e^{\left(\frac{E_{aHCI}}{kT}\right)}. \tag{2.1}$$

where T is the run-time temperature, α_{SA} is the activity factor, E_{aHCI} is the apparent activation energy, I_{sub} is the substrate current under stress at $V_G = V_D$, k is the Boltzmann's constant, f is the clock frequency and d_g is the transition delay.

2.1.2 Negative bias temperature instability

Unlike Hot Carrier Injection which affects both PMOS and NMOS transistors, the Negative Bias Temperature Instability (NBTI) affects only PMOS transistors. When the PMOS transistor is under inversion (the voltage at the gate terminal is at '0'), the hole induction breaks down the Si-H bonds and charge gets trapped in the gate oxide. This leads to an increase in threshold voltage (V_{th}) as well as a reduction in the drive current due to charge carrier mobility degradation. This shift in threshold voltage slows down the swithching speed of the transistor. Prior work shows that recovery of degraded parameters to a certain extent is possible when stress periods are followed by relaxation periods [3, 29, 43].

Since the NBTI stress occurs only when the PMOS transistor is in inversion, the amount of stress is dependent on the fraction of time for which the gate terminal of the transistor is held at a low voltage level. Based on the AC stress model for NBTI degradation proposed by Lu *et al.* [31], H.Kim *et al.* [25] have derived the lifetime of a single transistor as given below.

$$TTF_{NBTI} = \left[A_{NBTI}\left(\frac{1-\beta}{\beta}\right)^{n} e^{\left(\frac{nE_{aNBTI}}{kT}\right)}\right]^{1/n}. \tag{2.2}$$

where β is the dutycycle (fraction of time for which gate of the PMOS transistor is at V_{low}), E_{aNBTI} is the apparent activation energy, T is the run-time temperature, t is the operating time, k is Boltzmann's constant, n is the time exponent, and A is a fitting constant [31].

2.1.3 HCI and NBTI failure analysis

A single gate is usually considered to be over-aged when the threshold voltage change(ΔV_{th}) reaches 10% [43] and the equations (2.1) and (2.2) in the above sections can be used to determine the lifetime of a single gate. As it can be observed from these equations, the degradation because of HCI and NBTI happen under different operating conditions. While an increased activity in the circuit increases the amount of HCI stress, very low activity in the circuit may lead to biased duty cycles which will increase the amount of NBTI stress.

At first look, it may appear that the effect of usage is nearly opposite and decreasing one type of stress (among HCI and NBTI) can lead to increase in other. However, the activity factor α_{SA} *is not the inverse of duty cycle* β; when β is large, it is possible to make a substantial change to β without proportionally impacting α_{SA}. Furthermore, because of the $\frac{1}{(1-\beta)}$ term in Equation 2.2, 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).

2.1.4 Path delay

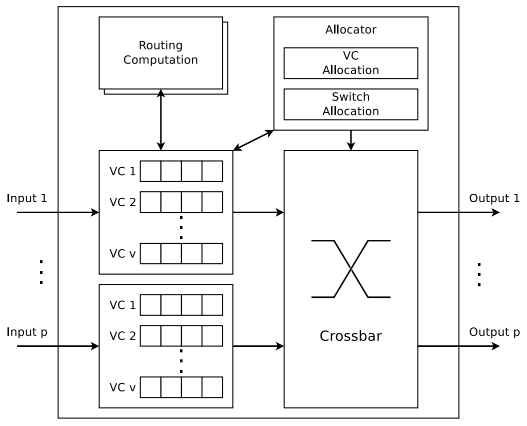
In the previous sections, we have examined the models which characterize the wear-induced transistor gate delays. However in complex digital designs, a single gate is part of a combinational path. Hence, in these digital systems, a path based delay model is needed to estimate the lifetime of the system. While all the paths will observe the wear-out, it is more likely that critical paths are affected first since the timing constraints are tighter on these paths. H.Kim *et al* [25] have proposed a model to calculate relative lifetime of a path between two latches, given the duty cycle of each gate along that path. The Acceleration Factor (AF) is defined as the ratio of the lifetime of the system under consideration, $T_{lifetime}(x)$, and a reference system, $T_{lifetime}(ref)$:

$$AF(x) = \frac{T_{lifetime}(x)}{T_{lifetime}(ref)} = \left(\frac{\sum_{j=0}^{M-1} \left(\frac{\beta_j}{1-\beta_j} \right)^n}{\sum_{i=0}^{N-1} \left(\frac{\beta_i}{1-\beta_i} \right)^n} \right)^{1/n} \tag{2.3}$$

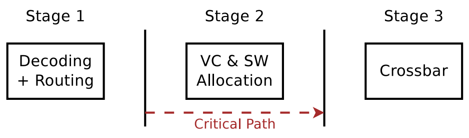
where β_i is the duty cycle of the i-th gate on the critical path of the system under consideration, and β_j is the duty cycle of the j-th gate on the critical path of the reference system. In Equation 2.3, it is assumed that the number of gates on the critical path of the two systems are N and M, respectively.We note that the method proposed here computes the relative lifetime improvement under NBTI degradation. As it will be discussed in Section 2.3, under the workloads examined, HCI degradation is low and relatively insensitive to incoming rate, thus its effect on lifetime was not modelled here.

2.2 Router microarchitecture

The canonical NoC virtual channel router is proposed by Peh and Dally [37]. Its block diagram is shown in Figure 2.1a. 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 2-D 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



(a)



(b)

Figure 2.1: Baseline router. (a) Router block diagram. (b) Router Pipeline Stages.

registers to keep track of their statuses, and buffers to store flits (flow-control units, 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., the 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.

The baseline router used by the prior work by H.Kim *et al* [25] is adapted from RTL code made publicly available by Becker [4], contains three pipeline stages. The three pipeline stages of this router architecture are shown in Figure 2.1b. Flit decoding and routing computation are done in *Stage 1*. The combined VC and switch

(SW) allocations are done in *Stage 2*. In *Stage 3*, flits traverse the crossbar.

As mentioned in the previous section 2.1, both HCI and NBTI slow down the transistor swtiching speed and the impact is first seen on the critical paths. H.Kim *et al.* [25] have synthesized the baseline router with 45nm TSMC standard cell library at 1 GHz and have determined the critical path of router. All the paths within 10% slack are considered as critical for this study and the results show that all the critical paths lie in the 2nd pipeline stage (VC and Switch allocation) of the router which is shown in Figure 2.1b. Simulations on the synthesized router with synthetic traffic show that the utilization of the allocators is closely related to the router incoming rate. Hence the stress time for HCI and NBTI which are closely related to the activity factor and duty cycle, will also be closely correlated to router utilization.

2.3 Workload characterization

H.Kim *et al.* [25] have studied the wear out of router circuits under synthetic and realistic workloads. The router described in Section 2.2 is synthesized using 45nm TSMC design library and all the paths with less than 10% slack were considered as critical paths for this wear characterization. As explained by the equations in Section 2.1, the amount of HCI and NBTI degradation are proportional to activity factor and duty cycles of nodes along these critical paths.

The post synthesis model of the router was stimulated with synthetic workloads with varying flit incoming rate and the corresponding effect on activity factor and duty cycles of critical path nodes was analysed. Figure 2.2 shows the impact of flit incoming rate on the activity factor of critical path nodes. At low incoming rates, there will not be much activity on these critical path nodes and hence the activity factor is very low. With out any prior knowledge one would expect the activity factor to increase with increasing incoming rates. While this is true to a certain extent, the

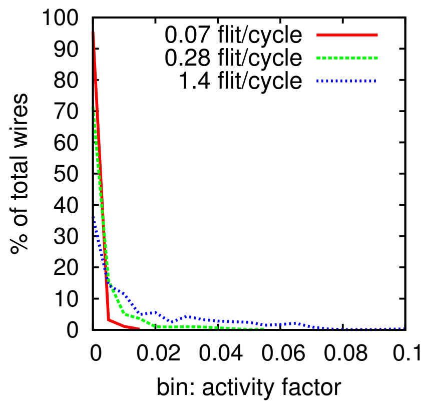


Figure 2.2: Activity factor with respect to router incoming rate. (H.Kim *et al.* [25])

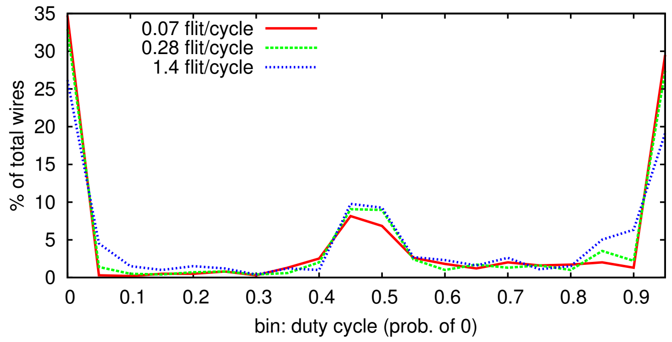


Figure 2.3: Histogram of duty cycle w.r.t incoming rate. (range = 0∼1.0, bin width = 0.05) (H.Kim *et al.* [25])

increase in activity factor is not very significant. As it can be observed, even at a very high incoming rate of 1.4 flits/cycle, the activity factor along these nodes does not exceed 0.1.

Similar study was done to study the impact of router incoming rate on the duty cycles along the critical path nodes, the results of which are shown in Figure 2.3. At low incoming rates, since there is lesser activity on these nodes, most of the nodes have a constant value of 0 or 1 and hence higher percentage of them have a duty cycle of 0% or 100%. At higher incoming rates, as the activity along these nodes

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

Table 2.1: System setup. (H.Kim *et al.* [25])

increases, there is an increase in nodes with duty cycles closer to 50%. But even at higher incoming rates, we can observe that significant fraction of nodes still have biased duty cycle of 100%. This is because most of the critical paths correspond to allocation corner cases in the router, which are very rare occurances.

The prior work has used PARSEC benchmarks to capture the incoming rates of routers under realistic workloads and also to study router-to-router variance of incoming rates. Gem5 simulator is used to run these parallel benchmarks and the system setup used to run these benchmarks is shown in Table 2.1. The results of this study are shown in Figure 2.4. The solid bars in the graph shows the average incoming rate among all the routers for each of these benchmark application and the router-to-router variance is shown by the whiskers. In Figure 2.4, the AVG shows the arithmetic mean of incoming rates for all the benchmarks and ALL shows the values for incoming rate when all the benchmarks applications are run one after the other sequentially. As it can be observed, the incoming rate is dependent on the the application and in general, the incoming rates for all the applications is very low (minimum of 0.0005 for *x264* and a maximum of 0.085 for *canneal*). Also, there is a huge variance in incoming rates among the routers for certain applications

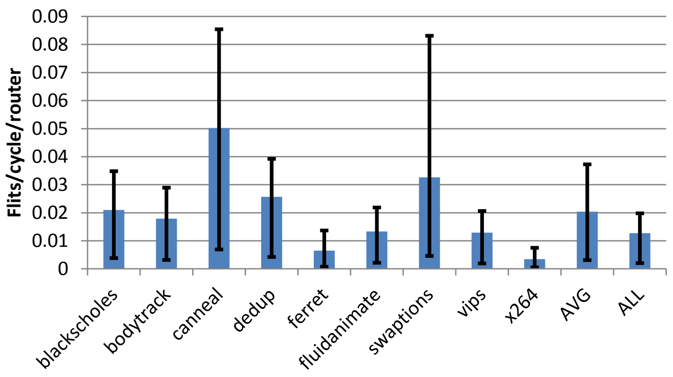


Figure 2.4: Average flit incoming rate per router for the entire range of the PAR-SEC benchmark suite (bars) and related range of incoming values (lines). (H.Kim *et al.* [25])

(*canneal* and *swaptions*) which shows that the router incoming rate also depends on the position of router in the 2D-mesh.

From the workload characterization of this prior work, the following conclusions can be made:

- HCI degradation is not very significant in the router circuits because of low activity factor among the critical path nodes even at high incoming rates.
- Because of biased duty cycles among the critical path nodes, NBTI degradation is dominant at low incoming rates.
- The realistic workload characterization shows that the incoming rates are usually very low and hence NBTI is the dominant transistor degradation mechanism in router circuits.

3. DETERMINISTIC LIFETIME EXTENDING ARCHITECTURE

The aging of transistors along the critical paths in a circuit depends on the work load causing the stress. As discussed in the previous chapter the work by H.Kim *et al.* [25] shows that the major degradation in Router circuits is caused by NBTI related stresses because of hugely biased duty cycles along the critical paths. Also the NBTI degradation is inversely proportional to duty cycle as described in chapter 2. To minimize the aging of transistors on these paths, H.Kim *et al.* [25] proposes to balance the duty-cycles of nodes on these paths by exercising the critical pipeline stage during idle periods with random inputs.

In this chapter, first we describe the wear-out resistant router architecture proposed by H.Kim *et al.* [25] and the limitations of the same in Section 3.1. Then we present a technique to maximize the lifetime improvement that can be obtained for this wear-out resistant router architecture in the next section.

3.1 Wear-out resistant router architecture

3.1.1 Approach

As discussed in Chapter 2, the previous work by H.Kim *et al.* have shown that the router circuits usually experience very low incoming rates under realistic workloads. Their analysis show that this low incoming rates lead to zero biased duty cycles which cause higher NBTI stress. The approach taken by H.Kim *et al.* to decrease this NBTI degradation is by exercising the critical paths of the router during idle periods of operation. While exercising the critical paths by artificially increasing the injection rate may improve the duty cycles of nodes along these paths, it might also increase the activity factor along these nodes and hence increase the HCI degradation and also the power consumption. However, the approach by H.Kim *et al.* have tried to improve the duty cycles with out a substantial impact on activity factor by *infrequently* changing the exercise mode input.

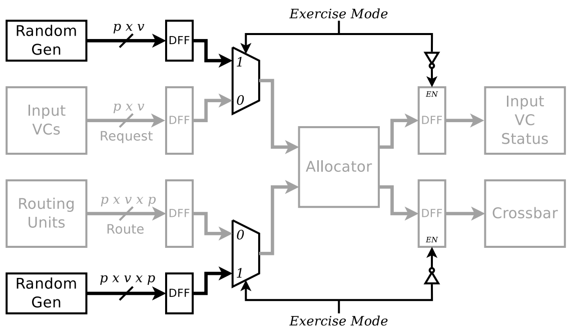


Figure 3.1: Exercise logic (original hardware in gray, proposed additional exercise logic in black). (H.Kim *et al.* [25])

Figure 3.1 illustrates the critical path of a baseline router, along with proposed modifications to reduce the wear-out effects of NBTI. The gates and wires in black are the additions to the baseline router. The critical paths in an NoC router, as stated in Section 2.2, lie within the Virtual Channel (VC) and crossbar allocation stages handled by the router allocator.

In the new micro-architecture shown in Figure 3.2, the “Random Gen” block is a ROM which stores pre-generated random vectors which will be used as exercise mode inputs. The exercise mode is turned on whenever the router allocator is idle and the exercise mode input vector replaces the “Request” and “Route” signals from the Input VCs. These random vectors are used to exercise the nodes along the critical paths which have biased duty cycles.

3.1.2 Limitations

While this approach might knock down the duty cycles of certain nodes along these paths, one cannot guarantee that nodes with highest duty cycle values are getting exercised. The nodes that get exercised by this approach completely depends on the random vectors that were being used as exercise mode input streams.

To improve on the technique presented by H.Kim *et al.* [25], we propose a method to deterministically generate a set of vectors which can be used as exercise mode input data. The objective of this deterministic generation of exercise mode data is to maximize the life-time improvement that can be achieved by exercise mode.

3.2 Maximizing the lifetime improvement

As mentioned in the above section 3.1.2, to get maximum possible benefit from the exercise mode, we need to make sure that all the nodes along the critical paths are exercised. To do this, we need a set of input vectors that are able to exercise all nodes on these paths. In order to generate such vectors, we need a complete characterization of logic cloud that affects the nodes on these paths. Once we have the logic characterization, we can use ATPG techniques to generate ideal set of exercise mode vectors which can exercise all the nodes on the critical paths.

3.2.1 Router critical path

Figure 3.2 shows a detailed view of 2^{nd} pipeline stage of the router (VC and SW allocation) in detail. As indicated by the dotted line in Figure 3.2, 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.

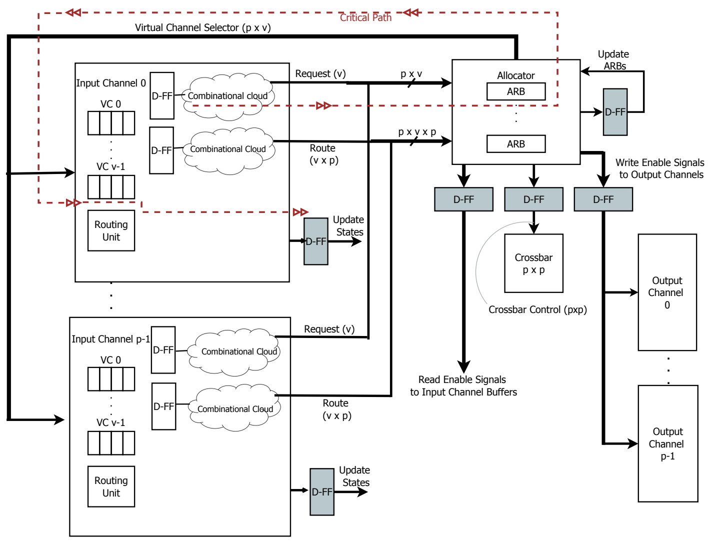


Figure 3.2: Virtual channel and switch allocation stage.

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 which VC 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 can be defined as a path from any of the primary inputs to any of the primary outputs. By starting at the end-point of a timing arc and building the logic cone backwards till 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 synthesis of 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%. Figure 3.3 shows the algorithm that we have used to extract this critical path logic.

Procedure Extract Logic Cloud ( )
Inputs: Baseline router netlist $\boldsymbol{R}$, critical timing paths $\boldsymbol{P}$
Outputs: Set of gates effecting critical timing paths $\boldsymbol{G}$, list of inputs to extracted logic cloud $\boldsymbol{PI}$

```
01: $\boldsymbol{G}$ = NULL;
02: $PI$ = NULL;
03: $\forall\ p_i \in \boldsymbol{P}$
04:    $Q = e_i$                          // $e_i$ is the end point of critical path $p_i$
05:                                       // $Q$ is a FIFO data structure.
06:    while ($\boldsymbol{Q} \neq \varnothing$)
07:       $x$'= pop ($Q$);  // Pop an element from queue.
08:       if ( isPrimaryInput($x$) )
09:          add $x$ in $\boldsymbol{PI}$
10:       else
11:          $g$ = getDriver($\boldsymbol{x}$)          // Get the driver gate of node $x$ from $\boldsymbol{R}$
12:          $\forall$ node $n_k \in$ fanIn($g$)
13:             if( isPrimaryInput($n_k$))
14:                add $n_k$ in $\boldsymbol{PI}$    // Add to primary inputs if not already present
15:             else
16:                push($Q$,$n_k$ )    // Add the node to the queue
17:             if( !isPresent($\boldsymbol{G}$,$g$) )// Add the gate to set of gates effecting critical timing paths
18:                add $g$ in $\boldsymbol{G}$
19: return $\boldsymbol{G}$, $\boldsymbol{PI}$;
```

Figure 3.3: Algorithm to extract logic affecting nodes on critical paths.

3.2.2 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 (*ie.* 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 logical circuit which forms each critical path. This critical path logic is extracted from the net-list generated by the synthesis of the router, as described above. The resultant combinational logic cloud has 1435 inputs and 357 outputs [*].

Figure 3.4 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 [†]. 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 chapter). 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 with a pre-defined 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 pre-defined rotation

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

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

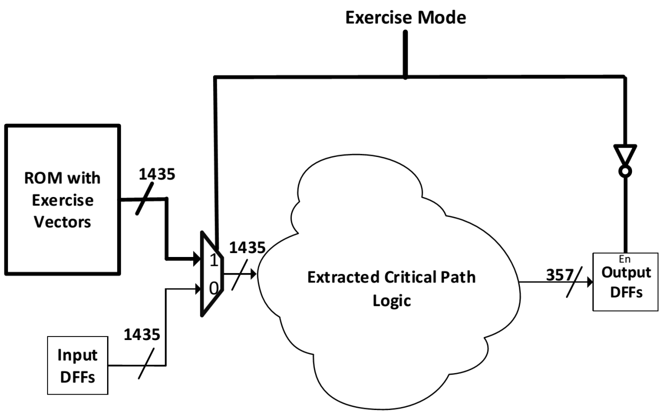


Figure 3.4: Critical path logic with proposed exercise logic. (additional exercise logic is darkened)

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 the period length on the circuit's energy consumption and lifetime in chapter 5. In the effort to minimize the impact on the router's timing, and hence the clock rate, the vector generation and all other exercise mode selection logic, as shown, are placed off the critical path.

4. VECTOR GENERATION

As mentioned in previous chapter 3, 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 [2] used for manufacturing tests for integrated circuits [11]. 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.

4.1 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 4.1a 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

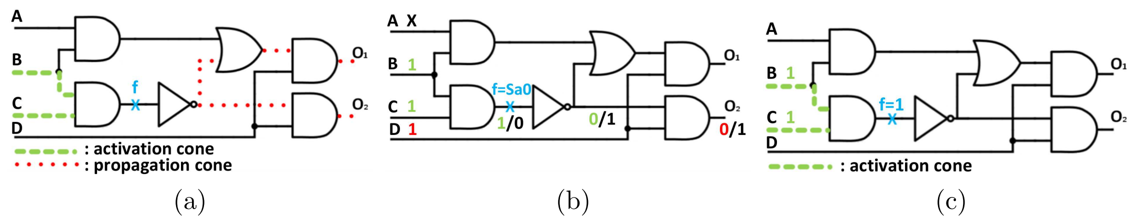


Figure 4.1: (a) Activation and propagation cones for fault location f; input signals B, C (A,B,C,D) determine activation (propagation), (b) test generation for f stuck-at-0; B=*1* and C=*1* activate the fault and D=*1* propagates its effect to O_2; possible test vectors $ABCD$=*X111* ={*0111, 1111*}, (c) let f be a critical net; exercising f=*1* requires activation of f stuck-at-0 with B=*1* and C=*1*.

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 4.1b 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 $\mathrm{v}_{ff}/\mathrm{v}_f$ 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 discussed above. 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 generate the necessary exercise vector. For example, it suffices to set *B=1* and *C=1* in Figure 4.1c in order to exercise signal f (which could belong to the critical netlist). The generated vector in this case is $ABCD=\{X11X\}$.

Possible MUX locations

Vector	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{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	I_1	I_5	I_6	I_9
v_1	X	1	1	0
v_2	0	X	1	1
v_3	1	0	0	1

(b)

Figure 4.2: (a) Possible ROM size 3×10 with 10 possible MUX locations, (b) necessary ROM size 3×4 with $4 + 4$ MUX locations.

4.2 Optimization of hardware overhead via compaction of exercise data

A considerable part of the hardware overhead of the exercise logic given in Figure 3.4 consists of the ROM which stores the exercise vectors as well as the various 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 4.2 which shows 3 exercise vectors. The row dimension of the matrix depends on the number of exercise vectors, 3 for 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 [11].

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 MUX'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 4.2a). 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 assumes either the constant *'0'* or 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 a MUX and a ROM column are needed as the value of the MUX data varies among different vectors (columns 1, 5, 6, 9 in Figure 4.2a). 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 above analysis the final ROM size in this example is (3×4). The number of necessary MUXes is the number of signals driven by a 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 vectors. Such an approach is described in the next section.

4.3 Generation of exercise vectors with large number of unspecified bits

The proposed vector generation algorithm is outlined in Figure 4.3. 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

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 $\boldsymbol{N}$ based on $\boldsymbol{D}$
02: $\boldsymbol{N}_e$ = NULL;                    // list of exercised critical nets
03: $N_{red}$ = NULL;                   // list of redundant critical nets
04: $j=1$;                          // exercise vector index
05: while ($\boldsymbol{N} \neq \varnothing$)
06:     $v_j = X$;                        // initialize $v_j$ with all unassigned values (don't cares)
07:     $\forall$ critical net $n_i \in \boldsymbol{N}$        // for each net not exercised yet
08:         $v_j$'= justify ($\boldsymbol{R}$, $n_i$, $v_j$);   // justify additional values of $v_j$ in order to exercise $n_i$
09:         if ($v_j$' != NULL )
10:             add $n_i$ in $\boldsymbol{N}_e$ and delete $n_i$ from $\boldsymbol{N}$
11:             simulate $v_j$' on $\boldsymbol{R}$
12:             $\forall$ $n_k \in \boldsymbol{N}$                 // for each net not exercised yet
13:                 if ($n_k$ == 1)
14:                     add $n_k$ in $\boldsymbol{N}_e$ and delete $n_k$ from $\boldsymbol{N}$
15:             $v_j = v_j$';                   // update current vector
16:         else
17:             add $n_i$ to $N_{red}$ and delete $n_i$ from $\boldsymbol{N}$
18:     add $v_j$ in $\boldsymbol{V}$
19:     $j$++;
20: return $\boldsymbol{V}$, $\boldsymbol{N}_e$;
```

Figure 4.3: Deterministic vector generation algorithm.

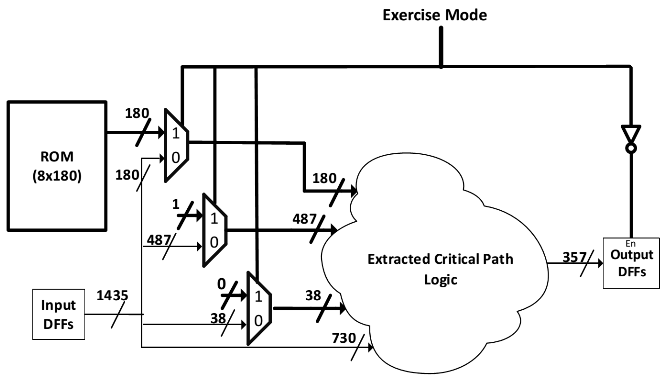


Figure 4.4: Critical path logic with proposed exercised logic (shown in bold), after vector generation.

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 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 05) 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. Firstly, 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 optimizing the hardware overhead via compaction of exercise data using the techniques discussed in Section 4.2. This is achieved by using a variant of a powerful in-house PODEM-based ATPG justification procedure [18], [33]. The justification procedure (line 08) 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.

4.4 Vector generation results and underlying exercise logic

Figure 4.4 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 3.2, and 14,653 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 4.3, *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 4.2 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 on Section 4.2) 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 4.4.

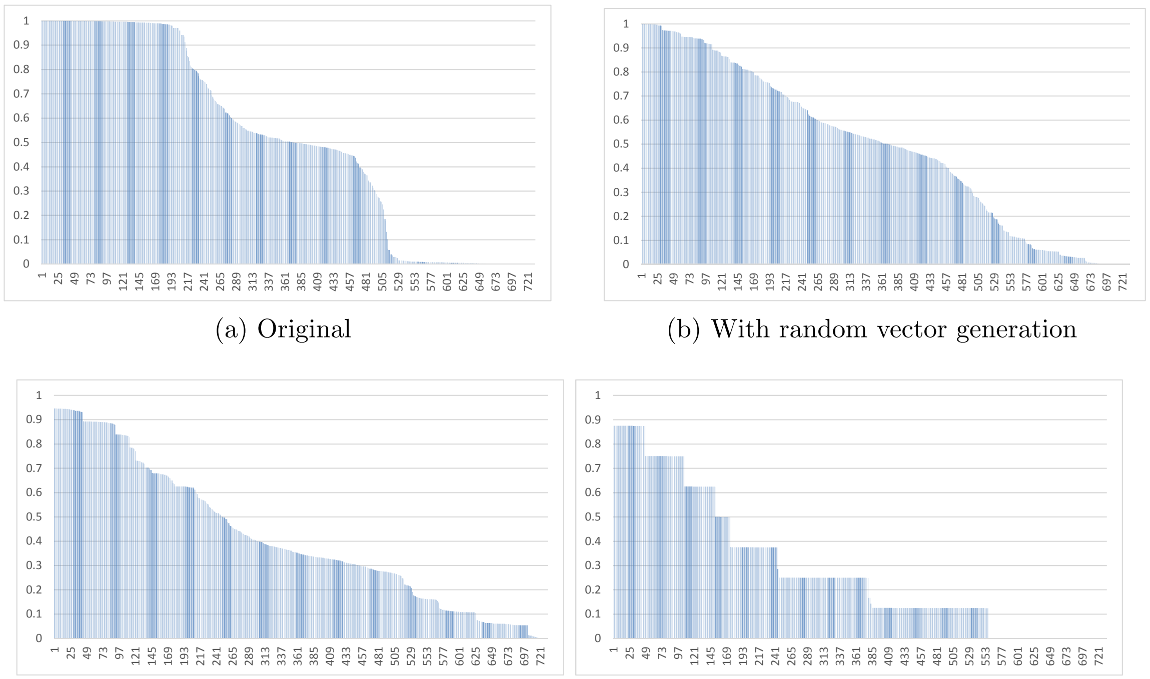
5. EVALUATION

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

5.1 Experimental setup

Since our work is an extension on top of the prior work done by H.Kim *et al.*, we have reused their experimental setup which would give us a platform to compare the advantages of our implementation. The three stage pipeline router was adapted from RTL code made publicly available by Becker [4]. As mentioned in Section 2.3, this router is synthesized using Synopsys Design Compiler mapped to a 45 nm technology library at 1 GHz. Even after the exercise mode additions to the baseline router, we were still able to synthesize at 1 GHz. This is because the additional circuitry is placed off the critical path. We have used other Synopsys tools like Design Vision to extract the critical paths within 10% slack, Primetime to estimate the power consumption. All the simulations of this post-synthesis router are done using VCS. We have developed in-house tools to extract activity factor, duty-cycles of individual nodes in the post-synthesis router. We have also developed an in-house tool to insert exercise mode multiplexors in the post-synthesis router netlist.

The router is evaluated under both synthetic and realistic workloads. The realistic workloads are captured as traces from gem5 [9] emulating a 64-core system executing multithreaded programs from the PARSEC v2.1 suite. Table 2.1 summarizes the system configuration. We compute 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 the realistic workloads like PARSEC benchmark programs. 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)

Figure 5.1: Duty cycles of critical path nodes with 2% incoming flit rate, sorted from highest to lowest.

5.2 Experimental results

5.2.1 Random versus deterministic vector generation

Aging due to NBTI depends on the duty cycles of nodes along the critical paths. We studied 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 5.1 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 5.1a shows, the duty cycles for 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 2.2. Figure 5.1b shows that using random vectors to exercise the critical paths produces improvement, but there are still a number of nodes with duty cycle of ~ 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 5.1d shows, the duty cycles improve greatly when the vectors used during exercise mode are generated using the deterministic method described in Section 4. After optimization, just 8 vectors, each 180-bits 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 5.1c, 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 5.1d.

5.2.2 Aging under synthetic workloads

We now examine the potential gain in router lifetime of the proposed technique versus baseline for a range of arbitrary incoming rates. Here, we also compare the

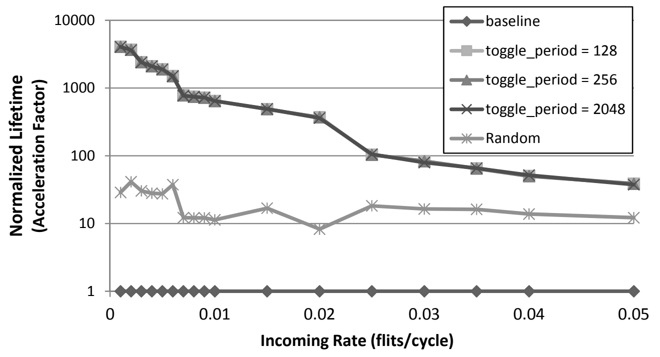


Figure 5.2: Normalized lifetime (acceleration factor) for router under a given synthetic incoming rate from 0.001 flits/cycle to .05 flits/cycle.

results obtained by using our deterministic method with the random inputs used by H.Kim *et al.* [25]. As previously discussed, the per-router incoming rate under PARSEC workloads varies between 0.0005 (*x264*) to 0.085 (`canneal`). Figure 5.2 shows the normalized acceleration factor (Equation 2.3) versus the baseline router at the same incoming rate. As explained in Section 2.1.4, the acceleration factor gives the lifetime of the system under consideration, normalized to the lifetime of the reference system. In Figure 5.2, "baseline" is the lifetime of the baseline router, normalized to 1, while "Random" indicates the case where the vectors in ROM are generated randomly (as in the prior work by H.Kim *et al.* [25]) and 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 5.2 shows. Generally low incoming rates cause a greater bias in duty cycle and also have more idle periods of operation, and hence, 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 with in 10% slack for our calculations. When random vectors are used to exercise the paths, the worst case path is different for different incoming packet rates. This will not happen if deterministic vectors are used, because an individual node will have the set of values under exercise mode.

Figure 5.2 shows no significant difference in lifetime between the three different vector rotation periods. This is because the duty cycle for a particular node on critical path will remain the same if 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 rotation period. For our simulation of 100,000 cycles and an incoming rate of 0.05 flits/cycle, 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 2048 for the remainder of this paper as this design point implies the lowest overhead in terms of activity factor.

5.2.3 Lifetime under PARSEC workloads

Figure 5.3 depicts the normalized lifetime of the network using the proposed technique under PARSEC workloads. For the lifetime estimation here, the router with lowest incoming rate for each benchmark is considered. This is because the router with lowest incoming rate will experience the highest NBTI stress. The acceleration factor in terms of lifetime here is computer for both random vector generation and deterministic generation with respect to baseline router recieving same incoming

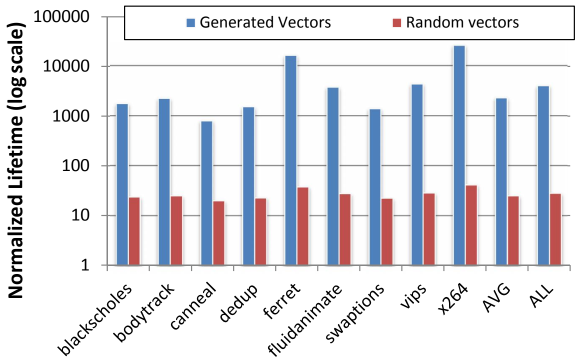


Figure 5.3: Normalized lifetime of the network using the proposed technique under realistic workload.

rate.

Deterministic vector generation achieves an average of ∼2300× reduction in wear rate (bars marked "AVG") as compared to that of random vector generation which only gives ∼28× improvement. As expected, both our proposed technique which uses deterministic generation as well as random vector generation performs better when incoming rate is low. Figure 2.4 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`), deterministic vector generation still achieves the normalized lifetime of 800× due to the extreme spread in per-router 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 ∼ 4000×. We found 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.

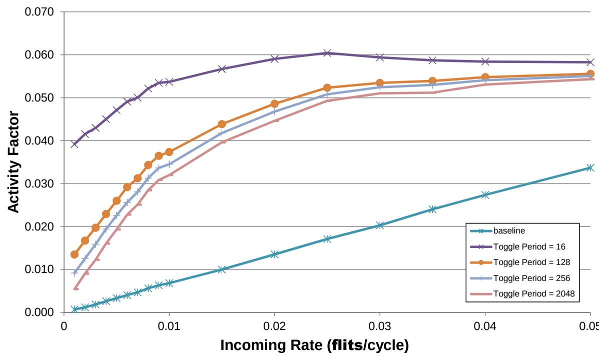


Figure 5.4: Activity factor versus injection rate.

5.2.4 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 to potential HCI-induced aging problems. Figure 5.4 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 opportunity 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 5.4, the impact of rotation period on activity

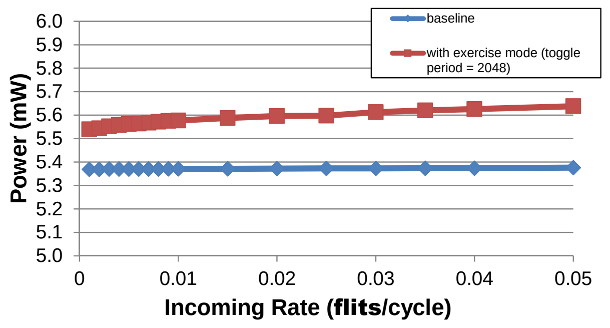


Figure 5.5: Router power consumption versus injection rate (note: Y-axis scaled to provide detail).

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

5.2.5 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 5.5 shows the power consumption with respect to varying incoming rate 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 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 baseline and router with exercise mode is from sequential circuit elements ($\sim 90\%$). During the exercise mode, only combinational circuit elements are switched, limiting the potential for power increase. Also the additional exercise mode logic increases router area by less than 5%.

6. RELATED WORK*

Our method described in the previous chapter provides an effective method to decrease the NBTI degradation in router circuits which are used in the interconnect of Chip-Multiprocessors. There are several techniques that have been proposed to mitigate NBTI aging effects for processor architectures. Colt duty cycle equalized proposed by Gunadi *et al.* [19] aims at balancing the duty cycles in the important components of processor like ALU by alternating between true and one's complement data representations. An NBTI aware processor called "Penelope" is proposed by Abella *et al.* [1] where they discuss different mechanisms to tackle NBTI in different processor components. For combinational blocks it is proposed that special inputs are executed during idle periods and for memory-like blocks they proposed mechanisms to write special values in empty entries of bit cells so that on average they will store 0 and 1 50% of the time. Kumar *et al.* [27] proposed periodic cache flipping so as to provide periods of relaxation for the influenced pMOSFET allowing dynamic recovery of the threshold voltage level.

Gupta *et al.* [21] proposed to generate 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. Stan *et al.* [20] proposed an approach to accelerate NBTI recovery by applying a negative supply voltage during idle periods of chip. In this case, the assumption is that the usage of systems follow a circadian rhythm and hence a period of usage will be followed by an extended period of idleness. Oboril and Tahoori [36] proposed to

*Part of this chapter is reprinted with permission from *Use it or lose it: wear-out and lifetime in future chip multiprocessors* by Kim, Hyungjun and Vitkovskiy, Arseniy and Gratz, Paul V and Soteriou, Vassos, in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, 2013*, pages 136-147, Copyright 2013 by ACM reduce aging in micro-processor 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. In another work, they proposed aging aware instruction set encoding called ArISE based on a heuristic optimization algorithm [34].

Next, Lai *et al.* [28] analyzed the effects of BTI on the clock distribution network with clock gating features in a microprocessor, 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 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.* [8] 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. However, the problem under investigation is proven to be NP-complete and hence the authors developed a linear-time heuristic that quickly produces good solutions; however, the solutions are only tractable for relatively small tree-structured digital circuits only.

All the prior works mentioned have some similarities to the method proposed by H.Kim *et al.* [25] in the sense that all of them propose to invert or use idle periods to reduce the impact of NBTI. These previous work however concentrate on the NBTI mitigation in single processor where are we concentrate on reducing the NBTI degradation in interconnect logic of Chip-multiprocessors. Moreover, all of the prior methods can be used only in the data path of the the processors. Our method concentrates on the control path of the router circuits which comprises the critical path in these circuits.

Aging has been also examined in the NoC domain. Bharadwaj *et al.* [6, 5] have proposed an adaptive routing algorithm 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. The 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 H.Kim *et al.* [25] showed are not actually the routers likely to exhibit the most stress-related aging. Fu *et al.* [17] 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 necessary 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, H.Kim *et al.* [25] have proved that this is not the actual case. Through detailed, gate-level analysis, not found in earlier works, they demonstrated that the duty cycle becomes more skewed when the NoC router is actually under-utilized and not when it is highly- or over-utilized.

Ever since the VLSI process technology crossed the sub-micron border, there has been extensive study on the aging models for transistors with an emphasis on HCI and

NBTI [35]. Since it is easy to measure the degraded transistor parameters under DC stress, there are fairly accurate models for these degradation mechanisms under DC stress [3, 29, 32]. But, the DC stress models can not be used for realistic workloads which observe stress under AC stress conditions at high frequencies. Attempts to study and develop models for such conditions have been done by the prior work [24, 42, 35, 39].

Prior work by Wang *et al.* [43] have made an attempt to make a combined model for HCI and NBTI. Similar work was also done by Fang *et al.* [14] which propose a model to calculate the delay increase for a given circuit under stress and by Lorenz *et al.* [26] which gives a combined model to calculate threshold voltave change. Both these works are for a particular technology library which require experimental determination of several parameters for HCI and NBTI. Furthermore the proposed work by Fang *et al.* [14] only characterized a small subset of 45 nm standard cell library which are sufficient for benchmark circuits. Because of this we were not able to use the same in our work. Also, all the prior models mentioned here lack important details such as the measurement conditions, detailed explanation of parameters etc... and hence it is fairly challenging to use these models in context of micro-architecture.

7. CONCLUSIONS

The NoC interconnect is critical to the lifetime survival of the CMP system. In this work, we extended the novel wearout-decelerating scheme proposed by H.Kim *et al.* [25] 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 is generated deterministically for maximum impact. We subsequently show that the proposed design yields a $\sim$2300$\times$ increase in router lifetime because of reduced NBTI wearout. In this work we have used a simple ATPG algorithm to generate the exercise mode data. It might be further possible to use advanced ATPG techniques which can result in reduction of ROM size needed to store the exercise mode data. This technique of using ATPG techniques to generate appropriate exercise mode data can be used for any arbitrary circuits. The ideal case would be when this technique can be incorporated as an additional option in the synthesis step of ASIC design flow. Then the tool chain can automatically determine the aging sensitive paths and will be able to generate additional logic required for the mitigation of such degradation.

REFERENCES

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

[2] Miron Abramovici, Melvin A Breuer, and Arthur D Friedman. *Digital systems testing and testable design.* Wiley-IEEE Press, Hoboken, New Jersey, USA, 1994.

[3] JEDEC Solid State Technology Association. Failure Mechanisms and Models for Semiconductor Devices, JEP122G, http://www.jedec.org/sites/ default/files/-docs/JEP122G.pdf. Technical report, October 2011.

[4] Daniel U Becker. *Efficient Microarchitecture for Network-on-Chip Routers.* PhD thesis, Stanford University, Stanford, CA, USA., 2012.

[5] Kshitij Bhardwaj, Koushik Chakraborty, and Sanghamitra Roy. An milp-based aging-aware routing algorithm for nocs. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 326–331. EDA Consortium, 2012.

[6] Kshitij Bhardwaj, Koushik Chakraborty, and Sanghamitra Roy. Towards graceful aging degradation in nocs through an adaptive routing algorithm. In *Proceedings of the 49th Design Automation Conference (DAC)*, pages 382–391. ACM/EDAC/IEEE, 2012.

[7] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Pro-*

ceedings of the 17th international conference on Parallel architectures and compilation techniques, pages 72–81. ACM, 2008.

[8] David R Bild, Robert P Dick, and Gregory E Bok. Static nbti reduction using internal node control. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 17(4):45:1–45:30, 2012.

[9] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

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

[11] Michael Bushnell and Vishwani D Agrawal. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, volume 17. Springer Science & Business Media, New York, USA, 2000.

[12] William J Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the Design Automation Conference*, pages 684–689. IEEE, 2001.

[13] Andrew DeOrio, Kostantinos Aisopos, Valeria Bertacco, and Li-Shiuan Peh. DRAIN: Distributed recovery architecture for inaccessible nodes in multi-core chips. In *Proceedings of the 48th Design Automation Conference*, pages 912–917. ACM, 2011.

[14] Jianxin Fang and Sachin S Sapatnekar. Incorporating hot-carrier injection effects into timing analysis for large circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(12):2738–2751, 2014.

[15] David Fick, Andrew DeOrio, Gregory Chen, Valeria Bertacco, Dennis Sylvester, and David Blaauw. A highly resilient routing algorithm for fault-tolerant nocs. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 21–26. European Design and Automation Association, 2009.

[16] International Technology Roadmap for Semiconductors (ITRS). Process integration, devices, and structures (PIDS), http://www.itrs.net/itrs%201999-2014%20mtgs,%20presentations%20&%20links/2009itrs/2009chapters_2009tables/2009_pids.pdf. Technical report, 2009.

[17] Xin Fu, Tao Li, and José AB Fortes. Architecting reliable multi-core network-on-chip for small scale processing technology. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 111–120. IEEE, 2010.

[18] Prabhakar Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Transactions on Computers*, 100(3):215–222, 1981.

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

[20] Xinfei Guo, Wayne Burleson, and Mircea Stan. Modeling and experimental demonstration of accelerated self-healing techniques. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.

[21] Saket Gupta and Sachin S Sapatnekar. Employing circadian rhythms to enhance power and reliability. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18(3):38:1–38:23, 2013.

[22] Jason Howard, Saurabh Dighe, Sriram R Vangal, Gregory Ruhl, Nitin Borkar, Shailendra Jain, Vasantha Erraguntla, Michael Konow, Michael Riepen, Matthias Gries, et al. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *Journal of Solid-State Circuits*, 46(1):173–183, 2011.

[23] Lin Huang and Qiang Xu. 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*, pages 51–56. ACM, 2010.

[24] Ulya R Karpuzcu, Brian Greskamp, and Josep Torrellas. The BubbleWrap many-core: popping cores for sequential acceleration. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 447–458. IEEE, 2009.

[25] Hyungjun Kim, Arseniy Vitkovskiy, Paul V Gratz, and Vassos Soteriou. Use it or lose it: wear-out and lifetime in future chip multiprocessors. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 136–147. ACM, 2013.

[26] Veit B Kleeberger, Martin Barke, Christoph Werner, Doris Schmitt-Landsiedel, and Ulf Schlichtmann. A compact model for nbti degradation and recovery under

use-profile variations and its application to aging analysis of digital integrated circuits. *Microelectronics Reliability*, 54(6):1083–1089, 2014.

[27] Sanjay V Kumar, Chris H Kim, and Sachin S Sapatnekar. Impact of nbti on sram read stability and design for reliability. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED)*, pages 210–218. IEEE, 2006.

[28] Liangzhen Lai, Vishal Chandra, Robert Aitken, and Puneet Gupta. BTI-Gater: An Aging-Resilient Clock Gating Methodology. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(2):180–189, 2014.

[29] Xiaojun Li, Jin Qin, and Joseph B Bernstein. Compact modeling of mosfet wearout mechanisms for circuit-reliability simulation. *IEEE Transactions on Device and Materials Reliability*, 8(1):98–121, 2008.

[30] Yanjing Li, Samy Makar, and Subhasish Mitra. CASP: concurrent autonomous chip self-test using stored test patterns. In *Proceedings of the conference on Design, automation and test in Europe*, pages 885–890. ACM, 2008.

[31] Yinghai Lu, Li Shang, Hai Zhou, Hengliang Zhu, Fan Yang, and Xuan Zeng. Statistical reliability analysis under process variation and aging effects. In *Proceedings of the 46th Annual Design Automation Conference*, pages 514–519. ACM, 2009.

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

[33] Stelios N Neophytou and Maria K Michael. Test set generation with a large number of unspecified bits using static and dynamic techniques. *IEEE Transactions on Computers*, 59(3):301–316, 2010.

[34] Fabian Oboril and Mehdi Tahoori. Arise: Aging-aware instruction set encoding for lifetime improvement. In *Proceedings of the 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 207–212. IEEE, 2014.

[35] Fabian Oboril and Mehdi B Tahoori. ExtraTime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level. In *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 1–12. IEEE, 2012.

[36] Fabian Oboril and Mehdi B Tahoori. Aging-aware design of microprocessor instruction pipelines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(5):704–716, 2014.

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

[38] Michael D Powell, Arijit Biswas, Shantanu Gupta, and Shubhendu S Mukherjee. Architectural core salvaging in a multi-core processor for hard-error tolerance. *ACM SIGARCH Computer Architecture News*, 37(3):93–104, 2009.

[39] Kewal K Saluja, Shriram Vijayakumar, Warin Sootkaneung, and Xaingning Yang. NBTI degradation: A problem or a scare? In *Proceedings of the 21st International Conference on VLSI Design (VLSID)*, pages 137–142. IEEE, 2008.

[40] Timo Schonwald, Jochen Zimmermann, Oliver Bringmann, and Wolfgang Rosenstiel. 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)*, pages 527–534. IEEE, 2007.

[41] Jared C Smolens, Brian T Gold, James C Hoe, Babak Falsafi, and Ken Mai. Detecting emerging wearout faults. In *Proceedings of the 3rd Workshop on Silicon Errors in Logic-System Effects*, pages 23–29. IEEE, 2007.

[42] Bogdan Tudor, Joddy Wang, Zhaoping Chen, Robin Tan, Weidong Liu, and Frank Lee. An accurate and scalable mosfet aging model for circuit simulation. In *Proceedings of the 12th International Symposium on Quality Electronic Design (ISQED)*, pages 1–4. IEEE, 2011.

[43] Yao Wang, Sorin Cotofana, and Liang Fang. A unified aging model of nbti and hci degradation towards lifetime reliability management for nanoscale mosfet circuits. In *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 175–180. IEEE Computer Society, 2011.

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