Enabling System-Level Modeling of Variation-Induced Faults in Networks-on-Chip

Enabling System-Level Modeling of Variation-Induced Faults in Networks-on-Chips*

Konstantinos Aisopos[†§], Chia-Hsin Owen Chen[§], Li-Shiuan Peh[§]
[†]Princeton University [§]Massachusetts Institute of Technology
{kaisopos, owenhsin, peh}@ csail.mit.edu

ABSTRACT

Process Variation (PV) is increasingly threatening the reliability of Networks-on-Chips. Thus, various resilient router designs have been recently proposed and evaluated. However, these evaluations assume random fault distributions, which result in 52%-81% inaccuracy. We propose an accurate circuit-level fault-modeling tool, which can be plugged into any system-level NoC simulator, quantify the system-level impact of PV-induced faults at runtime, pinpoint fault-prone router components that should be protected, and accurately evaluate alternative resilient multi-core designs.

Categories and Subject Descriptors: B.8 [Hardware]: Performance and Reliability
General Terms: Reliability, Measurement
Keywords: fault modeling, variation, Networks-on-Chips

1. INTRODUCTION

As silicon technologies move into the nanometer regime, devices become increasingly unreliable due to process variation (PV) and runtime variation. With each technology generation halving transistor size and the inability of the fabrication process to scale its precision accordingly[1], variations in transistor dimensions result in different electrical characteristics for each transistor. In addition, runtime conditions (*e.g.*, temperature, power consumption) have an increasing effect on device operation, since smaller sized transistors are more susceptible to runtime variations. This variability results in unpredictable critical path delays, which can lead to timing violations and faults. Thus, silicon failures can affect critical hardware components such as the Network-on-Chip (NoC). Unfortunately, a single fault in the NoC causes packets to be dropped or become corrupted, resulting in incoherent caches and erroneous memory traffic, ultimately causing the entire chip to fail.

*The authors acknowledge the support of the Gigascale Systems Research Center and Interconnect Focus Center, research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

[1]The wavelength of light to pattern transistors cannot be decreased below 193nm with current technology[10]. Consequently, the semiconductor industry is fabricating 45nm CMOS devices using 193nm optical light.

DAC 2011, June 5-10, 2011, San Diego, California, USA.

Resilient NoCs [3, 4] explore novel designs to allow continuous operation in the face of faults. However, they are all evaluated using a simple fault model that injects random faults (at gate or link level), uniformly distributed in time of occurrence and spatial location. As we show in Section 3.2, random fault injection results in 52% to 81% inaccuracy in capturing the fault locations of variation-induced faults, thus system-level designers cannot accurately identify the fault-prone components and protect them accordingly.

A more accurate fault modeling methodology is Static Timing Analysis (STA), used by chip designers to identify the critical paths that are likely to fail. However, PV changes the order of critical paths: paths that were indicated as non-critical by STA may become faulty, while the paths indicated as critical may not cause timing violations. To demonstrate this, we synthesize a baseline router (Table 1 in Section 2.1) with Synopsys Design Compiler at 3.8GHz on a 45nm process, perform STA with Synopsys PrimeTime, and extract the 512 most critical paths (the paths with the minimum slack), shown in decreasing delay along the x-axis of Figure 1. Then, we perform 100 Monte Carlo simulations for each path, varying the gate L/W, V_{th}, oxide thickness based on variation distributions obtained from the foundry, and plot (y-axis) the probability that each path will not meet timing. First, we note that in order to have no faults, the design has to run at only 75% of the frequency that it was synthesized for (*i.e.*, 2.85 GHz). Second, we observe that many critical paths with higher slack (less critical, towards right) have a higher probability of timing violation than critical paths with lower slack (more critical, towards left). For instance, we observe that paths with slack around 2.92% have a higher probability of violating timing than paths with slack around 2.4%. That is because static timing libraries cannot identify the actual critical paths under PV, where additional delays strongly depend on the layout of the paths' standard cells and other circuit-level parameters. Throw in runtime variations in temperature and power, and STA deviates further from reality.

Full-system simulation of multi-core chips is critical for evaluation of resilient solutions early in the design cycle. As multi-cores rely on NoCs for on-chip communications, the infrastructure needs to account for NoC reliability. However, no prior works provide a model/tool that can enable system level modeling of faults for NoC-interconnected chips. Since the fault-prone router components cannot be accurately identified by randomly injecting gate-level faults, nor by capturing the components with critical paths statically at design time (via STA), an accurate *variation-aware* router fault model is vital for early-stage evaluation of resilient ideas under PV.

Nicopoulos *et al.* [15], were the first to evaluate NoC's susceptibility to PV effects with rigorous circuit analysis, but did not develop a fault model/tool, nor extensively explored the system-level impact of faults. Various fault models have been developed though

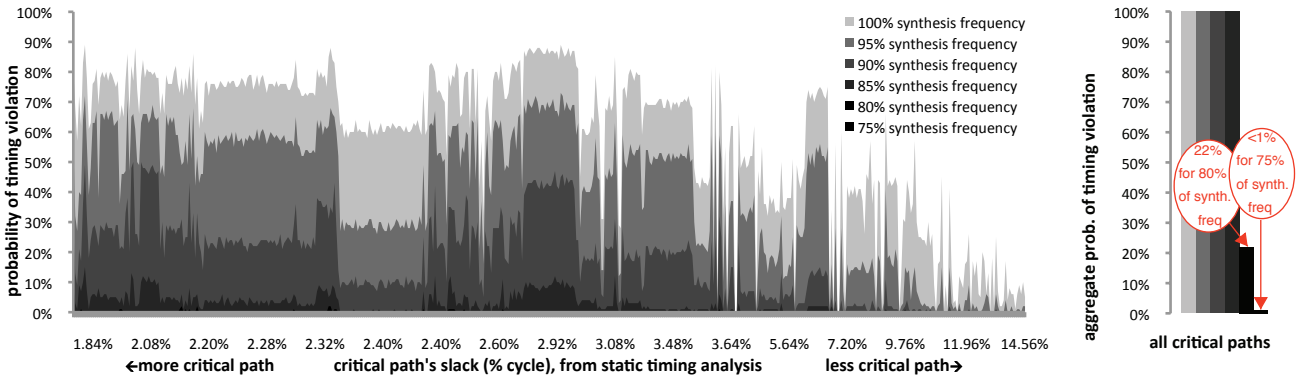


Figure 1: **Timing violation probabilities under Process Variation** (27°C, 512 most critical paths of router).

for memories and processors [11, 12, 16, 17]. These fault models typically depend on analytical modeling, which introduces inaccuracy[2] [16, 17] or project[3] future technology nodes [11, 12, 13].

We develop a router fault model based on the golden reference of the PV maps obtained from the foundry. As discussed in [18], SPICE models (and not the finished silicon) are the golden reference for the design of an ASIC, since once a technology has been defined and validated it is the job of the fab to produce silicon that matches this reference. We develop our fault model by synthesizing a highly parameterizable router RTL using 45nm technology, and performing Monte Carlo simulations directly on SPICE model netlists to capture timing violations due to variations of the manufacturing process. System-level designers only need to supply high-level NoC parameters to use our tool to explore the system-level impact of variation-induced timing violations.

Contributions. In this work, we make the following contributions:

- We develop a framework with circuit-level accuracy, that automates the mapping of timing violations to system-level faults, to explore the system-level outcome of variation-induced failures of NoCs (*e.g.*, misrouting, corrupted data, *etc.*). This framework captures the joint effect of PV and variation of runtime conditions (temperature and power consumption).
- We release a system-level fault modeling tool for NoCs, which can be plugged into any network simulator to inject variation-induced faults in routers' hardware at runtime, for designers to evaluate their resilient routers.
- We present a case study, characterizing faults at runtime, with or without fault-tolerant hardware in place, with GARNET [1], a NoC simulator that is part of the popular (over 1,700 users) system-level simulator GEMS [14] that models the OS, memory, and processors of multi-cores.

This paper is organized as follows: Section 2 describes how we develop a system-level fault model that captures the effects of variation without compromising circuit-level accuracy. In Section 3, we apply our fault model to characterize faults for various system configurations. Then, in Section 4, we present a case study where we demonstrate how our tool can be leveraged for evaluation of resilience at system-level. Finally, Section 5 concludes this paper.

[2]Analytical variation models can provide fault distributions very swiftly, since no actual simulation is required. However, they make assumptions to simplify complex equations that involve hundreds of physical parameters, significantly compromising accuracy; consequently, there is no widely accepted analytical model for determining the delay distributions under PV.

[3]Projecting a future technology with predictive models (*e.g.*, PTM [19]) is critical for early circuit design research, though its accuracy is limited. PTM adjusts 10 process/physical parameters (of about 100 in a BSIM model) and only considers their first-order correlations. The scaling factor and layout of standard cells is also set empirically based on previous technology nodes.

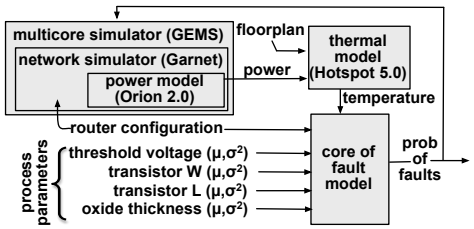


Figure 2: **Usage of fault model**.

2. PROPOSED NOC FAULT MODEL

The aim of our fault model is to detect when and where circuit-level timing violations will occur in NoCs and, for each potential timing violation, provide a system-level fault type in the form of a probability function. Though the core of our fault model is developed at circuit-level, the inputs and the outputs of the model are system-level variables (shown in bottom right of Figure 2). The inputs are the router configuration (number of input and output ports, number of virtual channels (VCs) or queues, number of buffers/VC, channel width, and operation frequency), the process parameters and their variations, and temperature. The output is the probability of occurrence for various system-level fault types. In order to capture the runtime temperature, a thermal model (*e.g.*, HOTSPOT 5.0 [6]) is required, which takes as input the floorplan and the runtime power of each router, provided by a network power model (*e.g.*, ORION 2.0 [8]). The power model obtains the run-time activity of NoC components from a network simulator (*e.g.*, GARNET [1]), whose traffic is driven by a multi-core simulator (*e.g.*, GEMS [14]).

To demonstrate potential ways that system designers can use our fault model for evaluating multi-core resiliency, we assume three hypothetical solutions that tackle the system-level fault types "data corruption", "flit[4] conservation", and "misrouting" (the first 3 fault types that our fault model detects, subsection 2.2, Table 2):

(a) *An Error-Correction-Code (ECC) that can tolerate corrupted bits in a packet's data*: our fault model provides the expected number of data bits that will be corrupted depending on the actual runtime conditions (traffic, power, temperature), so the network simulator can reflect these corrupted bits appropriately and various ECC policies (for instance Hamming codes) can be evaluated.

(b) *A coherence protocol that can tolerate lost and duplicate packets*: our fault model will allow the network simulator to duplicate or drop packets, thus a system-level resilient solution where the coherence protocol resends undelivered packets after a timeout, can be modeled in the multi-core simulator and faithfully evaluated.

(c) *A resilient routing algorithm*: our fault model will provide the probability of a packet being misrouted, which is reflected to

[4]Packets are broken into smaller pieces, called flits (**fl**ow control dig**its**).

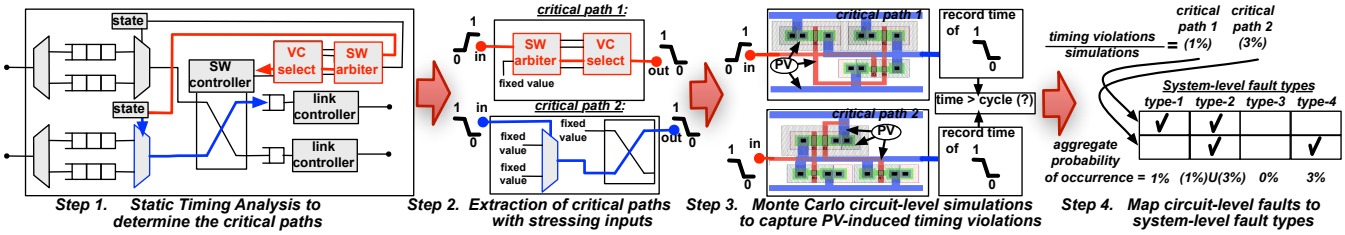


Figure 3: **Our fault modeling methodology**

the network simulator as a packet being sent to an incorrect output port at appropriate points in time. The effectiveness of a routing algorithm that can deliver the misrouted packets to their destinations without deadlocks can then be evaluated.

In Section 4, we present an actual case study with detailed simulation results, illustrating how the proposed fault model can be used for system-level evaluation of resilient multicores. The remaining of this section describes the methodology we use to develop the core of the fault model: we implement a highly-parameterizable router in RTL (subsection 2.1) and design the methodology of subsection 2.2 to measure the probability of faults for an experimental setup (a specific configuration of the high-parameterizable router and fixed temperature). Then, we perform a large number of experiments for a wide range of experimental setups, so we can observe the fault trends and create a database that captures the fault probabilities for a range of router configurations and runtime conditions.

2.1 Parameterizable Router Design

We implement in Verilog an input-buffered wormhole-switched router, with shared input buffers and credit-based flow control. The router design is similar to that of recent many-core chips, such as the 5GHz TERAFlops [5] and Intel's 2GHz 48-core chip [7]: each incoming flit gets buffered to an input buffer (BW), and then the output port to route to its destination is computed (RC). Next, it goes through switch allocation to reserve switch bandwidth (SA), selects a virtual channel (VA), traverses the switch (ST), and finally traverses the link (LT) onto the next router. We pipeline these functions into 4 stages: (BW/RC), (SA/VA), (ST), (LT), derived through synthesizing various component-to-stage assignments to obtain the most balanced timing across stages.

Our router design is highly parametrizable: it utilizes two virtual networks, one for data and one for control packets, for which the number of VCs, buffers/VC are parameterizable. Also, the number of input/output ports, channel width, and operation frequency are parameterizable. The nominal values are shown in Table 1.

data vnet	**num VCs**	2
	num buff/VC	3
control vnet	**num VCs**	2
	num buff/VC	1
channel width (bits)		64

num inputs	5 (4 directions, net interface)
num outputs	5 (4 directions, net interface)
operation frequency	80%, 75% synth frequency (3.04, 2.85 GHz)
temp, power	27°C, 0.2 watt

Table 1: Nominal router configuration (values are parametrizable)

2.2 Fault Modeling Methodology

In order to measure the fault probabilities for a specific configuration, under a fixed temperature, we need to perform a large number of Monte Carlo SPICE simulations for the router's layout, for variable V_{th}, gate dimensions, oxide thickness. However, chip-level simulation with timing is extremely time consuming and significantly limits the amount of input permutations and Monte Carlo simulations we can perform. On the other hand, STA has superior analysis speed and simulation completeness that no pattern set can provide, except that it is based on timing libraries that do not capture PV. Thus, we perform the following hybrid modeling procedure: we use STA to detect only the paths that are likely to fail, the worst-case input patterns that stress these paths, and then perform circuit-level simulation only for these paths and these input patterns. The steps of our methodology (Figure 3) are:

- *Step 1. Static Timing Analysis to determine the critical paths*
- *Step 2. Extraction of critical paths with stressing inputs*
- *Step 3. Monte Carlo circuit-level simulations of critical paths to capture PV-induced timing violations*
- *Step 4. Map circuit-level faults to system-level fault types*

Step 1. STA to determine the critical paths. In this step, the 256 slowest paths of the design, together with the input transitions that result in their longest delays, are recorded in a database. These are the paths that are most likely to fail[5] when PV in transistor dimensions and electrical characteristics results in additional timing delays for each cell. To achieve this, we perform STA with PrimeTime [18], using the timing library of IBM 45nm process. PrimeTime uses a vector-less approach that thoroughly explores the input space and all possible internal states of the synthesized netlist to identify the paths that can become critical at runtime.

Step 2. Extraction of critical paths with stressing input information. Once the critical paths have been identified, we extract their netlists, together with their worst-case input vectors. In the example of Figure 3, two critical paths are extracted; the first critical path is most likely to fail (worst input vector) when one input of the switch arbiter transitions from 0 to 1, while the other input maintains some fixed value (shown in Step 2). Each netlist contains a list of standard cells and their interconnections. To perform accurate circuit-level simulation, we extract the SPICE model for each cell of the critical paths from the IBM 45nm PDK and incorporate in our netlist all the parasitic elements resulting from the standard cells' geometry and internal wiring, including all parasitic capacitances and parasitic resistances.

Step 3. Monte Carlo circuit-level simulations of critical paths to capture PV-induced timing violations. The critical paths incorporating the SPICE models of cells are the golden reference of the most vulnerable paths of the chip. Next, we perform 100 Monte Carlo SPICE simulations for each critical path, at a specific input temperature, varying the process parameters (transistor dimensions, threshold voltage, gate oxide thickness) of its cells consistently with the statistical distributions provided by the foundry. The foundry assumes a Gaussian distribution for each process parameter and provides its mean (μ) and variance (σ^2). We note that we do not model spatial variation, since IBM 45nm SOI process claims that its intra-chip spatial variation is negligible and does not provide a model for it. In each simulation, we measure the delay of the critical path under the worst-case input vectors and mark whether the time till its output is ready remains shorter than the cycle duration, or if there is a timing violation. If there is a timing violation, we assume that the end register of the path will have

[5]Considering that only the 256 most critical paths (indicated by STA) are likely to fail is a fair assumption. As shown in Figure 1, only the 256 critical paths have a non-zero probability to violate timing for 75% or 80% of the synthesis frequency that we operate our baseline router design (Table 1).

	Vector of system-level fault types										
	data corruption		flit conserv.		misrouting		credit conserv.		erroneous allocation		unfair
critical path's end register / signal	few bits	all bits	dupli-cation	loss	wrong port	wrong vnet	gener-ation	loss	vc	switch	arbitration/ starvation
1. buffer ID where incoming flit is written		√		√			√	√			
2. switch request by SA winner		√		√	√					√	√
3. buffer ID where SA winner is to be read		√		√	√					√	√
4. vnet of SA winner						√					
5. destination VC of next router		√		√		√					
6. SA's outcome vector		√	√	√	√					√	√
7. SA's data buffer	√										
8. credit to be sent to previous node				√			√	√			
9. credit available in output buffer				√			√		√		√
10. head of free VC IDs queue		√		√			√	√	√		
11. free VC request by SA winner			√						√		√
12. ST's data buffer	√										

Table 2: Mapping the 12 most critical paths to system-level fault types

a random value in the next cycle, resulting in a fault. We define the fault probability of each critical path as the fraction of timing violations over the total number of simulations.

Step 4. Map circuit-level faults to system-level fault types. To capture the system-level impact of each failure, we design a framework to automate the mapping of timing violations to system-level faults. Each Verilog signal or register piggybacks a vector of system-level faults that will occur if it gets corrupted. The vector including all potential NoC system-level fault types is shown in Table 2, together with its values for the end registers/signals of the 12 most critical paths (of the 256 paths we are simulating). For instance, the first record corresponds to a timing violation of the path that generates the ID of the buffer where a received flit will be buffered. Upon such a failure the flit will be written to another random buffer (flit loss) potentially overwriting a useful flit latched there (data corruption). Also, since the buffer that the flit was intended for already sent a credit to the upstream router (indicating buffer availability for one flit) and not received the flit that used this credit, it assumes that the upstream router has not utilized this credit yet (credit loss). At system-level, the destination node will never receive the corresponding flit while, from now on, this buffer's capacity is reduced by one flit buffer. Note that our fault model only captures the first order effect of each failure; subsequent faults that may happen in later cycles as a result of the initial failure should be modeled in the network simulator. We note that if a fault is not masked by resilient hardware, it will eventually propagate to all stages of the router and potentially other routers as well.

After the fault probability of each critical path has been measured (Step 3), we set the probability of occurrence for each system-level fault type as the union of the fault probabilities of the critical paths whose failure triggers the corresponding system-level fault type (as indicated by the critical paths' vector of system-level faults). In Step 4 of Figure 3's example, the probability of failure is 1% for the first and 3% for the second critical path, thus the type-2 system-level fault type, which can be caused by either critical path failing, is $(1\%) \cup (3\%) = 100\% - [(100\% - 1\%) * (100\% - 3\%)] = 3.97\%$. We assume that the probability of each path failing is independent.

3. ROUTER FAULT CHARACTERIZATION

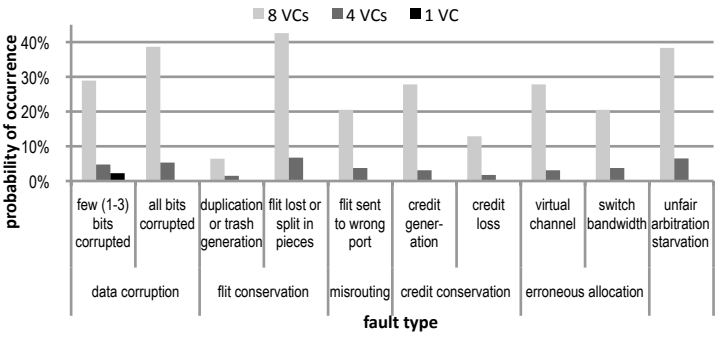
This section uses our model to better understand the fault locations and characteristics of NoCs. We first identify the exact locations where faults occur in each pipeline stage (subsection 3.1) and highlight their system-level impact. Using the fault locations that our model indicated as a reference point, we characterize the inaccuracy introduced by fault models with random distributions, and point out the importance of circuit-level modeling (subsection 3.2).

3.1 Fault Locations for Varying Configurations

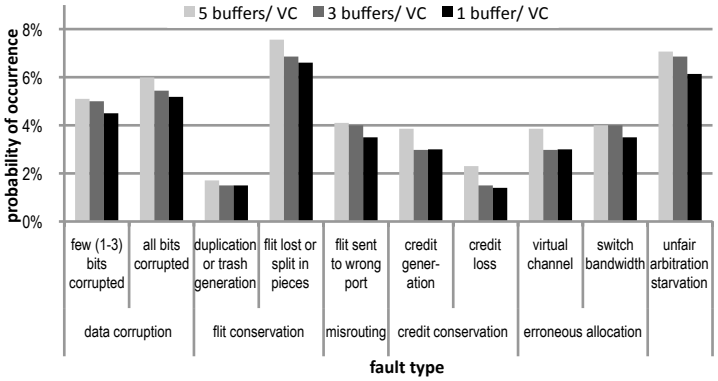
In order to generate a decent amount of PV-induced faults to study the fault-prone pipeline stages, we set the operation frequency to 80% of the synthesis frequency (22% aggregate fault probability for nominal configuration as shown in Figure 1), and set the remaining configuration parameters in their nominal values (as in Table 1). We then measure the probability of occurrence for each system-level fault type (Step 4 of Section 2.2), by performing Monte Carlo SPICE simulations for each critical path of our baseline router. We observe that 15.4% of paths fail with this configuration: 3.1% of these paths belong to BW/RC stage, 12.2% belong to the SA/VA stage, and 0.1% to the ST stage. In Figure 4a, we show the probability of each system-level fault type for variable number of VCs. Since the SA/VA stage is the most vulnerable, increasing the number of VCs will result in more complex logic for this stage, thus more paths will not meet timing, resulting in increased fault probabilities. We also vary the number of buffers/VC (Figure 4b), to demonstrate that adding complexity to the BW/RC stage will have a smaller impact on the fault probabilities, since fewer paths of this stage are expected to violate timing. We note that faults that are caused by few critical paths (*e.g.*, flit duplication) tend to have a small probability of occurrence, while faults that are related to data bits failing (*e.g.*, corruption of few bits) have high probabilities, since each of the 64 data bits violating timing will result in the same fault.

3.2 Comparison to Random Fault Model

In this section, we compare our fault model to the most commonly used fault model for system-level simulation of resilient Networks-on-Chip [3, 4], which is uniform random fault distribution at gate level. We use the fault rate that our model predicted to estimate the expected number of faults, and then we uniformly distribute them across the chip to measure the inaccuracy that is introduced by such a distribution (per fault type). For a fair comparison, given that the most critical pipeline stage can be easily identified by STA, we uniformly distribute faults to SA/VA stage only. We also experiment with uniformly distributing the faults only to the 256 critical paths that were identified by STA (if fewer critical paths are considered, the case is even stronger for our fault model). Figure 5 shows the inaccuracy introduced by these approaches; 0% inaccuracy corresponds to the distribution matching our model's probability of occurrence, while 100% inaccuracy implies that the distribution either does not predict that the corresponding fault type will occur, or doubles its probability. We observe that uniformly distributing faults across the 256 critical paths introduces 52% inaccuracy (on average), while uniformly distributing faults across SA/VA hardware introduces 81% inaccuracy (on average).



(a) **Fault probabilities for varying number of VCs** (buffers/VC = 3)



(b) **Fault probabilities for varying number of buffers/VC** (VCs = 4)

Figure 4: **Fault probabilities per fault type** (256 critical paths).

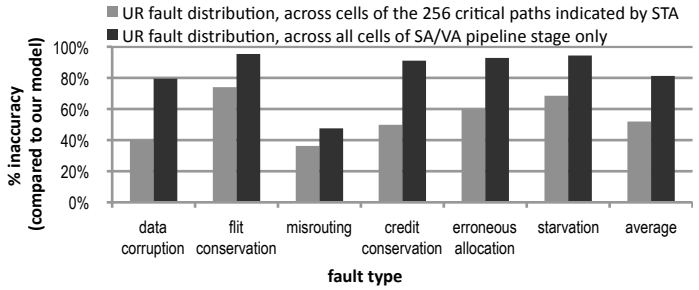


Figure 5: **Inaccuracy introduced by random fault models**.

This significant inaccuracy primarily stems from the fact that PV changes the order of critical paths (as motivated in our introductory section): paths that were not critical, but contain weak cells are more sensitive to variation in transistor dimensions and electrical characteristics, and thus are more susceptible to timing violations. In random fault models, the vulnerability of a critical path only depends on the gate count. On the other hand, our model takes into account the current that each gate requires to drive the following gates (as well as its fanout), the parasitic capacitances which result in additional undesirable delays, the voltage fluctuations due to threshold voltage variation, and various other interactions that can only be captured by circuit-level simulation. We frequently observe that paths that were not characterized as critical by our static timing analysis tool (PrimeTime), result in a significant number of timing violations in our Monte Carlo simulations. We conclude that, in the face of variations, a very large number of paths are potential timing violators and most of them cannot be identified by static timing libraries provided by the manufacturer, thus simulating the actual layout of the standard cells, as our model does, is critical for accurate modeling.

4. CASE STUDY: CAPTURING FAULT PROBABILITIES AT RUNTIME

In this section, we discuss a case study to showcase how our fault model can be used by system-level designers as a part of their multi-core and network simulators. To demonstrate this, we develop the simulation infrastructure detailed in subsection 4.1 and measure the expected runtime fault probabilities under synthetic traffic and PARSEC benchmarks (subsection 4.2). Then, we investigate the effect of fault tolerant system solutions (subsection 4.3).

4.1 System Configuration

We perform cycle-accurate full-system simulations with GEMS [14] for the 64-node system shown in Table 3. We set the power consumption of cores to 1watt (consistent with [5, 7]), and directly capture the runtime power for each router from the ORION 2.0 power model [8], which is integrated in GEMS's network simulator, GARNET [1]. At runtime, we feed these power numbers to HOTSPOT 5.0 [6] to estimate the temperature across the chip. Using the estimated temperature and power of each router, our fault model provides the fault probabilities on-the-fly. This simulation infrastructure was detailed in Section 2 and is shown in Figure 2 (of Section 2).

topology (for GARNET)	8x8 mesh with 4 memory controllers at the corners, router parameters as in Table 1
floorplan (for HOTSPOT)	256mm^2: 64 interconnected 2mm x 2mm nodes with 64 0.2mm x 0.2mm routers
L1 caching	32KB/node, private unified, 2 ways, MESI (latency: 3 cycles)
L2 caching	1MB/node, shared, distributed, 16 ways, MESI (latency: 15 cycles)
benchmarks	uniform random traffic, PARSEC suite (64 threads pinned to cores)

Table 3: System configuration

While in the previous section we set a high operation frequency to intentionally generate PV-induced faults, here we set the frequency to 75% of the synthesis frequency (close to 0% fault probability for nominal runtime conditions, Figure 1). This is because we assume that the chip manufacturer will opt to run the chip at a lower frequency to mitigate PV. So, this section explores the runtime faults, due to increased power and temperature, for a chip that was tested and shown to be fault-free under nominal conditions.

4.2 Fault Probabilities at Runtime

In order to demonstrate the effects of network traffic on fault probabilities, we warmup our network and then inject increasing uniform random traffic in intervals of 3,000 cycles (x-axis of Figure 6). Then, we plot the resulting temperatures of the warmest router, the coldest router, as well as average temperature of all routers, as shown in the top subfigure. For increased traffic, we observe high temperatures that exceed 100°C. This is a result of integrating state-of-the-art router designs in many-core chips with today's packaging technology. Recent research on on-chip router temperature profiling indicates that state-of-the-art router designs with standard packaging can develop temperatures up to 125°C under high traffic [2]. Such high temperatures result in 4% -10% fault probabilities (bottom subfigure of Figure 6) for a chip that is fault-free in nominal conditions. On the other hand, today's workloads are designed to scale up to small number of core counts, thus rarely put so much pressure in the network. When experimenting with PARSEC suite, we observe that the peak fault probability of the warmest router is 1% or below (Figure 7).

4.3 Fault Probabilities w Resilient Hardware

Recent research in system-level router design has explored various flavors of fault tolerant hardware to protect fault-prone components [3, 4]. Here, we model two simple techniques to protect the

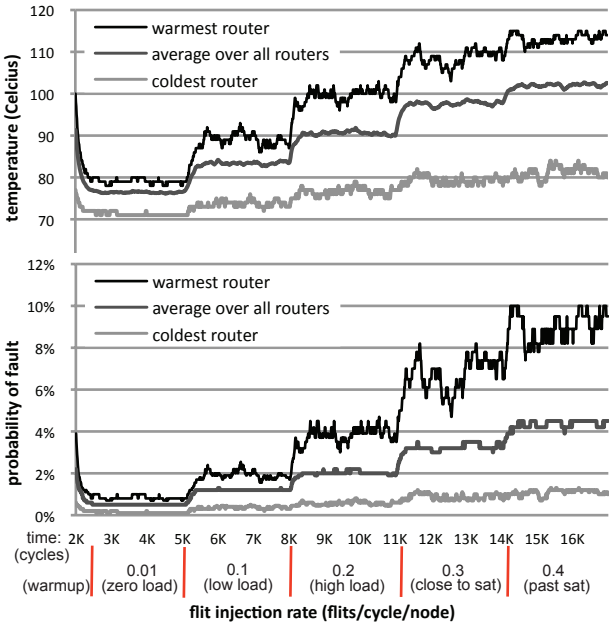


Figure 6: **Runtime fault probabilities** for increasing synthetic traffic

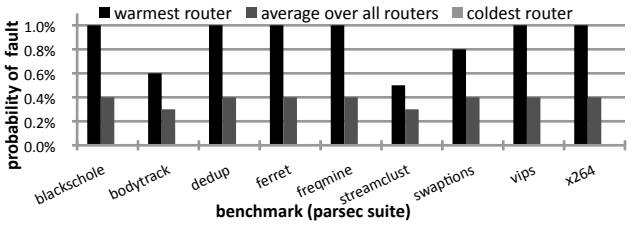


Figure 7: **Peak/average runtime fault probabilities** for PARSEC.

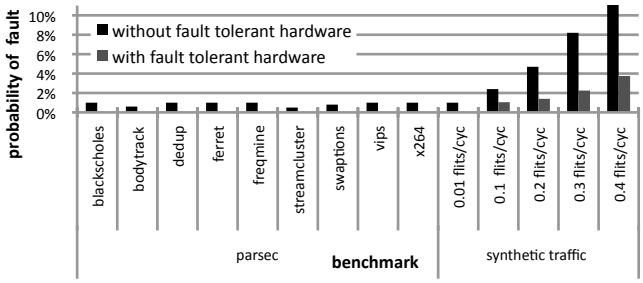


Figure 8: **Average runtime fault probabilities**, with and without resilient hardware, for synthetic traffic and PARSEC.

most vulnerable hardware components of our router (as identified by our characterization). We incorporate an Error Correction Code (ECC) to protect the flits from data corruption (we assume a Hamming code that can correct 1 to 3 bits whose value has been flipped). We also utilize triple modular redundancy to allocate the switch, where 3 identical structures perform switch allocation and the allocation winners are determined by voting logic that implements what the majority of the structures suggested. Figure 8 shows the average runtime fault probabilities with and without these resilient structures. We observe that resilient hardware results in fault probabilities close to zero for PARSEC benchmarks and low uniform random synthetic traffic, while for high synthetic traffic it decreases fault probabilities up to 6.8%

5. CONCLUSIONS

We have presented a fault modeling tool that can be easily integrated in system-level network simulators to capture runtime PV-induced faults in the NoC. Though the core of our model is implemented in circuit-level to capture accurate variation measurements, its system-level interface provides the probability of occurrence for system-level faults. We hope that this tool will ease the exploration of resilient NoCs and help researchers accurately evaluate their systems in the face of variation. Our future work includes exploring spatial variation and validating against silicon.

6. ACKNOWLEDGMENTS

We thank Nigel Drego for his guidance during the initial phase of this project. Nigel advised us to combine static timing analysis tools with circuit-level tools in our simulation infrastructure. Tushar Krishna designed our baseline RTL, based on [9].

References

[1] N. Agarwal, T. Krishna, L.-S. Peh, and N. Jha. GARNET: A Detailed On-Chip Network Model Inside a Full-System Simulator. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 33 –42, 2009.

[2] C.-H. Chao, K.-Y. Jheng, H.-Y. Wang, J.-C. Wu, and A.-Y. Wu. Traffic- and Thermal-Aware Run-Time Thermal Management Scheme for 3D NoC Systems. In *NOCS '10: Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pages 223–230, 2010.

[3] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky. BulletProof: A Defect Tolerant CMP Switch Architecture. In *International Symposium on High-Performance Computer Architectures*, 2006.

[4] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester. Vicis: a Reliable Network for Unreliable Silicon. In *Proceedings of the 46th Annual Design Automation Conference*, pages 812–817, New York, NY, USA, 2009.

[5] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. A 5-GHz Mesh Interconnect for a Teraflops Processor. *IEEE Micro*, 27(5):51–61, 2007.

[6] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, and M. R. Stan. HotSpot: Thermal Modeling for CMOS VLSI Systems. In *IEEE Transactions on Component Packaging and Manufacturing Technology*, 2005.

[7] J. Howard et al. A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS. *International Solid-State Circuits Conference*, pages 108–109, 2010.

[8] A. Kahng, B. Li, L.-S. Peh, and K. Samadi. ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 423–428, 2009.

[9] T. Krishna, J. Postman, C. Edmonds, L.-S. Peh, and P. Chiang. SWIFT: A SWing-reduced Interconnect For a Token-based Network-on-Chip in 90nm CMOS. In *Proceedings of International Conference on Computer Design*, 2010.

[10] S. Kundu, A. Sreedhar, and A. Sanyal. Forbidden Pitches in Sub-Wavelength Lithography and Their Implications on Design. *Journal of Computer Aided Materials Design*, 14(1):79–89, 2007.

[11] X. Liang and D. Brooks. Microarchitecture Parameter Selection to Optimize System Performance Under Process Variation. In *International Conference on Computer-Aided Design (ICCAD)*, pages 429–436, 2006.

[12] X. Liang and D. Brooks. Mitigating the Impact of Process Variations on Processor Register Files and Execution Units. In *Proceedings of the annual IEEE/ACM International Symposium on Microarchitecture*, pages 504–514, 2006.

[13] Y. Lu, L. Shang, H. Zhou, H. Zhu, F. Yang, and X. Zeng. Statistical Reliability Analysis Under Process Variation and Aging Effects. In *Proceedings of the 46th Annual Design Automation Conference*, pages 514–519, New York, 2009.

[14] M. Martin et al. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *Computer Architecture News (CAN)*, 2005.

[15] C. Nicopoulos, S. Srinivasan, A. Yanamandra, D. Park, V. Narayanan, C. R. Das, and M. J. Irwin. On the effects of process variation in network-on-chip architectures. *IEEE Transactions on Dependable and Secure Computing*, 7:240–254, 2010.

[16] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects. *IEEE Transactions on Semiconductor Manufacturing*, 21(1):3 –13, 2008.

[17] S. R. Sarangi, B. Greskamp, and J. Torrellas. A Model for Timing Errors in Processors with Parameter Variation. In *ISQED*, pages 647–654, 2007.

[18] T. Thiel. Have I Really Met Timing? – Validating PrimeTime Timing Reports with Spice. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, Washington, DC, USA, 2004. IEEE Computer Society.

[19] W. Zhao and Y. Cao. New Generation of Predictive Technology Model for Sub-45nm Design Exploration. In *ISQED '06: Proceedings of the 7th International Symposium on Quality Electronic Design*, 2006.