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

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1145/2024724.2024931">http://dx.doi.org/10.1145/2024724.2024931</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Association for Computing Machinery (ACM)</td>
</tr>
<tr>
<td>Version</td>
<td>Author's final manuscript</td>
</tr>
<tr>
<td>Accessed</td>
<td>Mon Oct 29 13:31:53 EDT 2018</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/72480">http://hdl.handle.net/1721.1/72480</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike 3.0</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/3.0/">http://creativecommons.org/licenses/by-nc-sa/3.0/</a></td>
</tr>
</tbody>
</table>
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, to 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, 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2011, June 5-10, 2011, San Diego, California, USA.
Copyright 2011 ACM ACM 978-1-4503-0636-2 ...$10.00.
Figure 1: Timing violation probabilities under Process Variation (27°C, 512 most critical paths of router).

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 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

---

2. 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, without or with 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.

---

4 Packets are broken into smaller pieces, called flits (flow control digits).
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 TERAFlips [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), 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 traversing various component-to-stage assignments to obtain the most balanced timing across stages.

Our router design is highly parameterizable: it utilizes two virtual networks, one for data and one for control packets, for which the number of VC, 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.

#### Table 1: Nominal router configuration (values are parametrizable)

<table>
<thead>
<tr>
<th>data vnet</th>
<th>num VCs</th>
<th>2</th>
<th>num inputs</th>
<th>5 (4 directions, net interface)</th>
</tr>
</thead>
<tbody>
<tr>
<td>control vnet</td>
<td>num VCs</td>
<td>2</td>
<td>num outputs</td>
<td>5 (4 directions, net interface)</td>
</tr>
<tr>
<td></td>
<td>num buffer/VC</td>
<td>1</td>
<td>operation frequency</td>
<td>100%, 75%ynthia frequency (3.04, 2.85 GHz)</td>
</tr>
<tr>
<td>channel width (bits)</td>
<td>64</td>
<td>temp, power</td>
<td>27° C, 0.2 watt</td>
<td></td>
</tr>
</tbody>
</table>

### 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 VC, 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 when PV in transistor dimensions and electrical characteristics result 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 inputs

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 when PV in transistor dimensions and electrical characteristics result in additional timing delays for each cell. 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

\[ \text{time} > \text{cycle} \]

#### Step 4. Map circuit-level faults to system-level fault types

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 TERAFlips [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 traversing various component-to-stage assignments to obtain the most balanced timing across stages.

Our router design is highly parameterizable: it utilizes two virtual networks, one for data and one for control packets, for which the number of VC, 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.

<table>
<thead>
<tr>
<th>data vnet</th>
<th>num VCs</th>
<th>2</th>
<th>num inputs</th>
<th>5 (4 directions, net interface)</th>
</tr>
</thead>
<tbody>
<tr>
<td>control vnet</td>
<td>num VCs</td>
<td>2</td>
<td>num outputs</td>
<td>5 (4 directions, net interface)</td>
</tr>
<tr>
<td></td>
<td>num buffer/VC</td>
<td>1</td>
<td>operation frequency</td>
<td>100%, 75%ynthia frequency (3.04, 2.85 GHz)</td>
</tr>
<tr>
<td>channel width (bits)</td>
<td>64</td>
<td>temp, power</td>
<td>27° C, 0.2 watt</td>
<td></td>
</tr>
</tbody>
</table>

### 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 VC, 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 when PV in transistor dimensions and electrical characteristics result 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 inputs

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 when PV in transistor dimensions and electrical characteristics result in additional timing delays for each cell. 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
Table 2: Mapping the 12 most critical paths to system-level fault types

<table>
<thead>
<tr>
<th>Critical path's end register / signal</th>
<th>Data corruption few bits</th>
<th>Flit conservation loss</th>
<th>Misrouting wrong port</th>
<th>Wrong vnet</th>
<th>Credit conservation loss</th>
<th>Errorous allocation vc</th>
<th>Unfair arbitration / starvation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Buffer ID where incoming flit is written</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2. Switch request by SA winner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3. Buffer ID where SA winner is to be read</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4. Vnet of SA winner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5. Destination VC of next router</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6. SA's outcome vector</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>7. SA's data buffer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>8. Credit to be sent to previous node</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>9. Credit available in output buffer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>10. Head of free VC ID's queue</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>11. Free VC request by SA winner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>12. ST's data buffer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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 a small number of very critical paths (e.g., bit 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).

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).
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 1 watt (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).

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
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


