Sthira: A Formal Approach to Minimize Voltage Guardbands under Variation in Networks-on-Chip for Energy Efficiency

Raghavendra Pradyumna Pothukuchi, Amin Ansari,* Bhargava Gopireddy, and Josep Torrellas
University of Illinois at Urbana-Champaign **Qualcomm Research*
http://iacoma.cs.uiuc.edu

***Abstract*—Networks-on-Chip (NoCs) in chip multiprocessors are prone to within-die process variation as they span the whole chip. To tolerate variation, their voltages (V_{dd}) carry over-provisioned guardbands. As a result, prior work has proposed to save energy by operating at reduced V_{dd} while occasionally suffering and fixing errors. Unfortunately, these proposals use heuristic controller designs that provide no error bounds guarantees.**

In this work, we develop a scheme that dynamically minimizes the V_{dd} of groups of routers in a variation-prone NoC using formal control-theoretic methods. The scheme, called *Sthira*, saves substantial energy while guaranteeing the stability and convergence of error rates. We also enhance the scheme with a low-cost secondary network that retransmits erroneous packets for higher energy efficiency. The enhanced scheme is called *Sthira+*. We evaluate Sthira and Sthira+ with simulations of NoCs with 64–100 routers. In an NoC with 8 routers per V_{dd} domain, our schemes reduce the average energy consumption of the NoC by 27%; in a futuristic NoC with one router per V_{dd} domain, Sthira+ and Sthira reduce the average energy consumption by 36% and 32%, respectively. The performance impact is negligible. These are significant savings over the state-of-the-art. We conclude that formal control is essential, and that the cheaper Sthira is more cost-effective than Sthira+.

***Keywords*-Network on chip; Variation; Control theory.**

I. Introduction

Variations in process, supply voltage (V_{dd}), and temperature parameters present an increasingly important challenge to achieve energy efficiency in processors. These variations arise from the reduced voltage and feature dimensions in upcoming processors [1]. To build resilient processors that can tolerate such variations, designers are forced to use conservative guardbands, defeating the goal of energy efficiency.

Networks on Chip (NoCs) are especially prone to such variations. They connect distant parts of the chip which, due to variations, exhibit different characteristics. Hence, the NoC has to be designed to safely work in the slowest and in the leakiest parts of the chip. This leads to conservative designs and energy inefficiencies. Unfortunately, the NoC consumes a substantial fraction of the on-chip energy — potentially up to ≈30%, according to the literature [2], [3], [4], [5], [6], [7] — and this contribution may increase in communication-dominated future exascale systems [8]. As a result, we need to find novel NoC solutions that balance the opposing goals of low energy and variation tolerance.

A possible approach to achieve energy efficiency in a variation-affected environment is to operate the design with reduced guardbands, occasionally suffering and fixing errors. Prior proposals such as Razor [9] and BlueShift [10] have used circuit techniques to run cores at high clock frequencies for the available timing guardbands. Tangle [11] has reduced the V_{dd} guardbands of routers in an NoC without changing their frequency. Bacha and Teodorescu have reduced the V_{dd} guardbands in processors while monitoring ECC feedback [12], [13].

Nearly all of these proposals, however, use *ad hoc* decisions, relying heavily on empirically-tuned settings to control the frequency or the V_{dd} that leads to improved operation. Such heuristic algorithms are often too conservative, and sacrifice energy efficiency. Alternatively, they are sometimes highly complex, which makes their design, tuning and verification efforts prohibitively high. Moreover, the settings of the ad hoc schemes are known to work only for the specific environments that are used to design them, and we do not know how well they work under other conditions not encountered at design time. Lastly, ad hoc schemes do not have a clear design methodology. This means that reusing the design for future architectures requires a full search of a large design space because the current design is highly tuned to the current architecture.

To address these limitations, we require *formal* methodologies. In this paper, we use formal control techniques to design a V_{dd} controller. Our goal is to dynamically control the V_{dd} of groups of routers in a variation-prone NoC, keeping V_{dd} as low as possible while sustaining a small but tolerable number of errors. The design, called *Sthira*, provides guarantees on the convergence, stability, and maximum resulting error rates for the NoC under control. The result is a robust, scalable and energy-efficient system.

We propose two Sthira designs: a basic version and an aggressive variant called *Sthira+*. The latter additionally includes a low-cost secondary network that operates at nominal V_{dd} and retransmits a packet when it has suffered an error in the primary NoC. Retransmitting the packet on the primary NoC could result in repeated errors, which would then force the controller to increase the steady-state V_{dd} suboptimally.

We evaluate Sthira+ and Sthira with simulations of variation-affected NoCs with 64–100 routers running a mix of workloads. With 8 routers per V_{dd} domain, our schemes reduce the average energy consumption of the NoC by 27% with negligible performance overhead. For a futuristic design with one router per V_{dd} domain, Sthira+ and Sthira reduce the average energy consumption of the NoC by 36% and 32%, respectively, with negligible performance impact. These savings are obtained after taking into account the penalty of power losses in voltage regulators, and are substantially higher than those attained by prior approaches. While the secondary network helps Sthira+ attain higher energy savings, its non-negligible hardware cost makes Sthira+ less cost-effective than Sthira.

Overall, the contributions of this work are:

- The design of Sthira, a scheme that dynamically minimizes the V_{dd} of groups of routers in an NoC using formal control-theory approaches.

- Sthira's enhancement into Sthira+, a scheme that further integrates a low-cost secondary network operating at nominal V_{dd} for higher energy efficiency.
- An evaluation of Sthira and Sthira+ that demonstrates large and controllable energy reductions with negligible performance impact over the best existing approaches.

II. Background

A. Process Variations and Energy Savings

With decreasing feature sizes, process variations have become an important concern for chip manufacturers [1]. In this paper, we focus on Within-Die (WID) process variations. WID variations have a systematic and a random component [14], [15]. The random component is due to random dopant fluctuations, while the systematic component is typically due to imprecisions in the manufacturing process, and exhibits significant spatial correlation. WID variations may reduce the delay in some paths and increase it in others. This results in higher V_{dd} guardbands and lower operating frequencies at the chip level, as dictated by the slowest paths.

Aggressively reducing V_{dd} or frequency guardbands decreases energy consumption but can create timing violations, as some of the variation-affected paths may be too slow. If such violations can be detected and corrected without significant overheads, overall energy efficiency can improve. This insight has been used by several researchers to develop heuristics at the circuit level [9], [10], [16] or at the architectural level [11], [12], [13], [17] to reduce energy consumption in a variation-affected chip. Specifically, Razor [9] and BlueShift [10] decrease the frequency guardbands in processor pipelines. Tangle [11] and Bacha and Teodorescu [12], [13] reduce V_{dd} guardbands in NoCs and processors while keeping the frequency unchanged. Hi-ECC [17] saves energy by reducing DRAM refresh frequency at the expense of increasing the strength of ECC codes. Unlike these works, we propose a control-theoretic scheme to reduce the V_{dd} of the NoC routers (without changing their frequency). Additionally, we observe that we can obtain large energy savings by tolerating a small but non-zero sustained error rate, instead of completely avoiding the errors. Therefore, our design lowers the V_{dd} of the routers to sustain a constant error rate.

B. Modeling Timing Faults

As V_{dd} decreases, certain paths may start missing timing under certain logic values and cause intermittent faults. This timing fault rate increases as V_{dd} decreases. VARIUS-NTV [15] is a tool that models process variations and the timing violations that ensue from reduced guardbands. VARIUS-NTV takes a certain logic structure (e.g., the synthesized RTL implementation of an NoC router), and gives the probability of a timing error for each path in the different pipeline stages, for a given V_{dd}. In this paper, we conservatively assume that whenever a timing violation occurs, it causes an incorrect execution.

C. Supporting Multiple V_{dd} Domains

An effective approach to tolerate process variations is to divide a logic component into multiple V_{dd} domains. In this way, a high V_{dd} can be applied to sections of the component that are slow due to systematic variations, and a low V_{dd} to sections that are fast. In an NoC, a natural V_{dd} domain is a set of neighboring routers.

Currently, using multiple on- or off-chip Switching Voltage Regulators (SVR) to provide multiple V_{dd} domains consumes significant area and power [18]. However, upcoming technology will provide better solutions. One approach may involve integrating V_{dd} regulators hierarchically [19], [20]. The first level of the hierarchy consists of a few SVRs on a stacked die or on the package, while the second level consist of many on-chip low-drop-out (LDO) regulators. Each LDO is fed by one of the SVRs and provides the V_{dd} for a domain. The area overhead of LDOs is negligible, as they reuse the hardware for a power-gating circuit. In addition, LDOs have high efficiencies if the ratio of their output (V_O) to input (V_I) voltage is close to 1. Also, level converters, required for communication across V_{dd} domains, can be efficiently designed by combining them with latches [21]. In this paper, we use 8 routers per V_{dd} domain for a conservative design, and 1 router per V_{dd} domain for an aggressive, futuristic design. The IBM POWER8 processor chip has demonstrated a hierarchical, LDO-based V_{dd} regulator system that attains a peak power efficiency of 90.5% [22]. Therefore, to keep our analysis simple, in this paper we assume that a near-future system that provides multiple V_{dd} domains in the NoC wastes 10% of the total power to voltage regulation.

III. Necessity of Formal Control

The general approach of designing controllers using heuristics has a few drawbacks. First, the effort to design and tune the controller is high, due to the lack of a consistent methodology. In addition, much of this effort has to be repeated when designing a slightly-different controller. What's worse, even after painstaking efforts, hand-tuned heuristic-based algorithms may incur unpredictable and unacceptable behavior outside the training set [23]. As a result, heuristic schemes are typically kept simple to avoid such unpredictable behavior. With a few exceptions, this makes the algorithms conservative and reduces their capability to adjust their actions to the dynamic execution.

The alternative is to use control-theoretic techniques [24]. Such techniques are formally derived, and manage the system in an organized manner. They gracefully adapt to changes, and provide guarantees on: (i) convergence, (ii) stability, and (iii) error bounds [23]. Convergence means that the system will eventually reach the desired operating point specified by the designer — if such an operating point exists. Stability means that once the steady state is attained, the system will not deviate from it. Error bounds guarantees limit the maximum error that the system can have once the system reaches the steady state. They also limit the peak overshoot and undershoot while in the process of reaching the steady state. Overall, control-theoretic systems can deliver aggressive benefits without unpredictable behavior.

As an example of the need for formal control, consider the heuristic scheme used in Tangle [11] to reduce energy in the context of process variations. A controller periodically reduces the V_{dd} of the routers in an NoC and, if an error occurs in a flit, immediately raises the V_{dd} of all routers in the path of that flit. This approach saves some energy in the NoC, but is conservative in its decision-making and sub-optimal.

Figure 1 shows an execution timeline where the V_{dd} of a router is changed by a heuristic controller like Tangle's when running GemsFDTD, a SPEC06 benchmark. The details of the experimental infrastructure are presented in Section VI. The same figure also shows the changes on the same router driven by a formal controller that we propose in this paper.

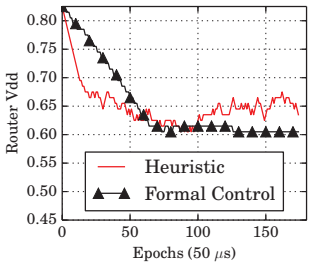


Figure 1: V_{dd} of a router as tuned by a heuristic scheme or by a formal-control scheme.

The desired operating point here is the lowest V_{dd} at which the router operates with a pre-specified tiny error rate. Both schemes initially reduce the router V_{dd} to a low value. However, under the heuristic scheme, the V_{dd} slowly sways away from it. It makes many abrupt changes, but does not return to the desired setting again. This approach cannot guarantee that the router's V_{dd} will converge to the desired value. Additionally, it does not guarantee stability — i.e., after the V_{dd} reaches the desired value, it slowly moves away or makes large oscillations around it.

On the other hand, under the formal scheme, the V_{dd} stays close to the low value. It may oscillate a bit, but the controller provides the three guarantees defined above.

The formal scheme also allows the desired timing error rate to be specified statically or dynamically. This reference specification controls the tradeoff between the amount of energy saved and the performance overhead incurred. In some scenarios, high energy-efficiency is paramount; in others, only minimal errors can be tolerated. The controller updates the router's V_{dd} to achieve this error rate. In addition, due to its guarantees, it is assured that the actual error rate will be within the allowed bounds.

IV. Sthira Architecture

Sthira's goal is to dynamically reduce the V_{dd} of groups of routers in a variation-prone NoC (without changing the frequency) in a way that guarantees the convergence and stability of V_{dd}, and the bounds on flit error rates. For this, Sthira uses a controller developed using a formal, control-theoretic approach. The controller uses light-weight control connections to all the routers to read the error rate in the routers and update the routers' V_{dd}. We also present a more advanced design called Sthira+ that, in addition, adds a low-cost secondary network to save additional energy.

In Sthira, the controller changes the V_{dd} at periodic intervals called *Epochs*. The V_{dd} is updated only once per epoch — at the end of the epoch, irrespective of the number of flit errors observed. The length of the epochs is adaptive, in that if there are only a few flits in the network, the controller postpones any V_{dd} tuning on a router until enough flits have been seen by the router to make a decision. When Sthira increases the V_{dd}, it does so only on the router (or group of routers) that logged the error.

This section describes the formal controller and the secondary network. The discussion implicitly assumes the availability of a V_{dd} domain per router. However, our evaluation will assess both a conservative NoC design with 8 routers per V_{dd} domain, and a futuristic one with one router per V_{dd} domain.

A. Formal Controller

1) Modeling the Error Rate: The design of a formal controller requires a model of the relationship between the error rate of a router and the router's V_{dd}. Figure 2 shows the probability of a timing error occurring in an NoC router as a function of the router's V_{dd}, as obtained from studies of an NoC with 64 routers using VARIUS-NTV and other tools (described in Section VI).

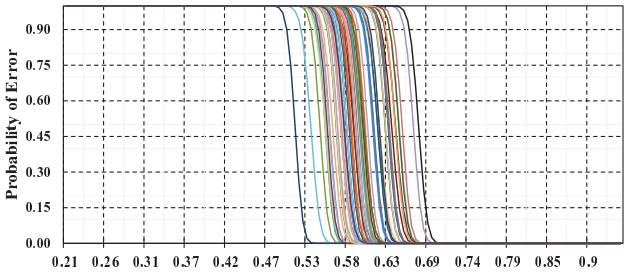


Figure 2: Probability of a timing error as a function of V_{dd} for different routers in a 64-router NoC. Each curve corresponds to one router.

The figure shows a steep increase in the probability of an error in a router as the V_{dd} applied is reduced. Due to process variations, the minimum error-free V_{dd} for different routers is spread across a wide range. For V_{dd} values above $720mV$, all the routers are free of failure; the chance of failure for any router is less than 10^{-18} failures per cycle. However, only a few routers can operate at a V_{dd} as low as $570mV$ without failure.

In this graph, each curve has three distinct regions: the two extremes and the central region. At the extremes of each curve, the V_{dd} is either so low that the probability of failure is 1, or it is high enough that the probability is 0. The central region of the curve covers a large range of error rates. We want the routers to operate only in the very lowest part of the central region, near negligible error rates.

We need to model the "S" shaped error probability curves mathematically, and then use the model to design a controller that works for the very low segment of the central region. A straightforward approach is to build a separate model and a controller for each of the curves. However, this is impractical. Therefore, we take advantage of the fact that all the curves in Figure 2 have a similar structure. Hence, instead of building a model for each router separately, we build a statistical model that works for all the routers.

The nominal value of this statistical model at each V_{dd} is obtained by computing the mean value of all the curves at that V_{dd}. The standard deviation of the model includes all the curves in Figure 2. This stochastic model can now represent the error probability distribution of all the routers in the chip. Also, being a statistical model, it also represents

curves that are slightly different from those in the figure. These extra curves include shifts or tilts of the curves in Figure 2. With this approach, we simplify model building, and improve the generality of the model to capture slightly different behaviors that may occur in deployment.

We try to model the "S" shaped error probability function in Figure 2 using different sigmoid functions, such as the Error Function (*erf(x)*), Logistic Distribution, and Gompertz function. More details on the properties of these functions and their applicability for sigmoid distributions can be found in [25], [26], [27], [28]. We find that the Error Function *erf(x)*, shown in Eq. (1), fits the error probability curve closely. Using this function, we obtain Eq. (2) as our statistical model, which gives the probability of an error as a function of the router's V_{dd}. Since our desired operating point is at the very bottom of the central region, we locally approximate the function in Eq. (2) in that region using a first-order Taylor polynomial, and obtain Eq. (3). We validate the accuracy of this approximation and design our controller using standard techniques [29].

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-t^2} \mathrm{dt} \tag{1}$$

$$P(Error) = 0.5(1 - erf\left(\frac{V_{dd} - 0.595}{0.012}\right) + erf\left(-\frac{V_{dd} - 0.595}{0.012}\right)) \tag{2}$$

$$\widehat{P(Error)} = 0.5 - \frac{1}{\sqrt{\pi}}\left(\frac{V_{dd} - 0.595}{0.012}\right) \tag{3}$$

When a router is operating outside the modeled region (e.g., at system start-up), a module brings the V_{dd} to the modeled region. At that point, the controller takes over.

2) Controller Design: The foremost consideration in the design of the controller is that the controller should stabilize the system at the specified operating point. The controller should not cause sudden or large increases in the error rate. Moreover, the controller should work correctly despite potential modeling errors. Lastly, the controller overhead should be kept small. With the above considerations, we implement our controller as a discrete-time closed-loop PID (Proportional, Integral, Differential) controller with output feedback. PID controllers have been widely used in several domains due to their flexibility and wide range of applicability [29]. The PID controller takes the error rate of a router and the reference error rate ($E_\circ$) as inputs and generates a change in the V_{dd} of that router. The output $\Delta V_n(i)$, which is the change from the current V_{dd} for the n^{th} router in the i^{th} epoch, is given (in simplified form) as:

$$\Delta V_n(i) = K_P \Delta E_n(i) + K_I \sum_{k=0}^{i} \Delta E_n(k) + K_D\{\Delta E_n(i) - \Delta E_n(i-1)\} \tag{4}$$

where $\Delta E_n(i) = E_n(i) - E_\circ$, and $E_n(i)$ is the error rate for the n^{th} router in the i^{th} epoch. The first term in Eq. 4 represents the Proportional Gain (G_P), i.e., the change in V_{dd} proportional to the deviation of $E_n(i)$ from $E_\circ$. The second term represents the Integral Gain (G_I), i.e., the change in V_{dd} proportional to the sum of all the deviations that occurred before. The last term represents the Differential Gain (G_D), i.e., the change in V_{dd} proportional to the rate at which the deviation has changed from the previous epoch to the current epoch. K_P, K_I, and K_D are the proportionality constants in these terms.

For a general system, using a simple controller with Proportional Gain alone would result in a steady-state error. Therefore, we add an Integral Gain that can nullify this error. However, the integral component tracks only the history and may not be responsive enough for large sudden deviations (e.g., an increase in error rate). To incorporate a timely reaction to large sudden deviations and use a predictive action, we also add a Differential Gain. This gain addresses rapid changes in error rate, reducing overshoots and keeping the error rate close to the reference value.

The controller generates changes in V_{dd} to get close to the requested error rate. As the system gets closer to the steady state, the V_{dd} changes become progressively smaller. However, arbitrarily small V_{dd} steps are not be possible due to limitations of the voltage regulators (VRs). As a result, we restrict the minimum magnitude of the V_{dd} step that the controller produces to be the minimum tuning step of the VR, called V_{MS} (e.g., $10mV$). So, the output of the controller is rounded to the nearest available VR step. For example -2 mV is rounded to 0, while +6 mV is rounded to 10 mV.

Figure 3 illustrates the overall design. The NoC is divided into several V_{dd} domains, each consisting of a group of routers. The PID controller is time-shared across all the domains. When it is a domain's turn, the controller considers each of its routers, one at a time. It monitors a router's error rate and computes a desirable V_{dd} change. After all the routers in the domain have been analyzed, the overall V_{dd} change for the entire domain is decided, and passed to the VR that is connected to that domain. The VR adjusts the V_{dd} of the domain. More details are given in Section V-B. Figure 3 shows the controller generating a decision for domain *j*.

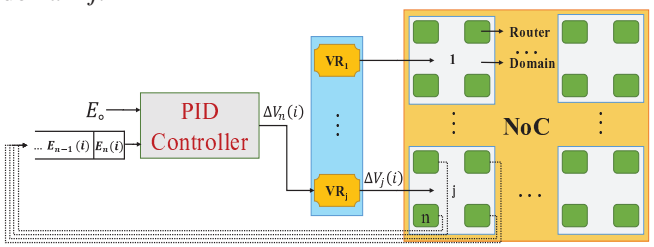


Figure 3: Overview of Sthira.

To design the controller, we use the z-transform [29] of the difference equation of error rate as a function of V_{dd}, and tune the PID controller using MATLAB. Using the root-locus method to stabilize the controller, we obtain a range of values for K_P, K_I and K_D. We start with a settling time of 35 epochs and target the peak overshoot to be 0.1% error rate. We then tune K_P, K_I and K_D to minimize the deviations around the reference error rate after the controller enters the stability band (after the settling time). In this band, we define the acceptable error margin to be 5% of the reference error rate. Using pole-zero analysis to keep the settling time, overshoot and undershoot acceptable, we find the values of K_P, K_I and K_D to be -2.9966$\times 10^{-3}$,-4.9766$\times 10^{-6}$ and -7.1804$\times 10^{-2}$. It is possible to choose

a different set of values at design time if a different convergence time is desired. To counter potential modeling errors arising from model simplifications and/or hardware limitations such as the quantized VR output, we add a gain margin of 35% and a phase margin of 60° to the controller design. This means that the controller will work correctly even if the error rate magnitude for a given V_{dd} changes by up to $\pm$ 35%, or if the slope of the error rate vs. V_{dd} relationship as captured by the model changes by up to 60°. These margins are used in a standard manner in control theory to make PID controllers robust.

3) Hardware Implementation: The controller's hardware is shown in Figure 4. It performs a four-step operation at the end of an epoch. First, it calculates the deviation of a router's error rate from the reference value. The router's error rate is obtained by dividing the number of erroneous flits seen by the router by the total number of flits sent through the router. The division process is approximated using bit-shifts. Second, the hardware calculates the accumulated error rate deviation (σ_n) and the change of error rate deviation from the previous epoch (δ_n). This requires information from the previous epoch, which is stored in two tables indexed by the router number, n. Third, the values of the current deviation, accumulated deviation and change of deviation are used to obtain the values of the G_P, G_I, and G_D gains from three 128-entry tables of precomputed gains (similar to [30]). Lastly, the controller hardware adds the gains to generate the change in V_{dd} for that router. This change is quantized based on the minimum VR step. This process uses fixed-point adders and occurs only once per epoch. The very short latency of this process does not impact performance.

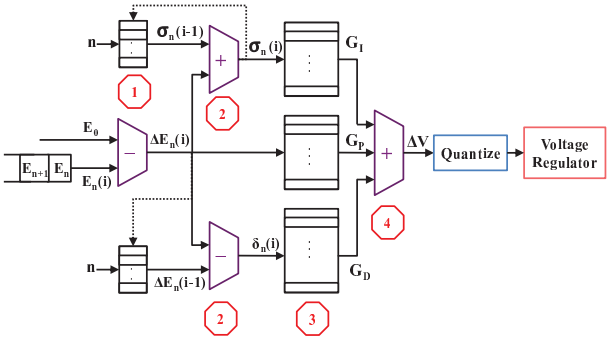


Figure 4: Controller implementation.

B. Sthira+: Low-Cost Secondary Network

Sthira does not increase the V_{dd} of a router immediately when an error is detected; the V_{dd} can only be increased at the end of an epoch. This is done to avoid over-reacting after an error. However, when the packet is retransmitted, since the V_{dd} conditions have not changed, depending on the state of the router, it is possible that the retransmitted packet ends up suffering an error as well. This process may slow down message delivery and may make the controller believe that the error rate is higher than it really is. As a result, the Sthira controller may end up setting the V_{dd} of the router to a value higher than needed.

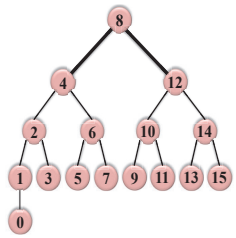
To avoid this problem, we also propose a more advanced Sthira design called *Sthira+* that adds a low-cost *Secondary Network* that operates at the nominal V_{dd} — and is, therefore, error free. When an error occurs, the packet is retransmitted through this secondary network. This guarantees immediate packet delivery and ensures high performance, albeit at the cost of an additional, simple network.

The purpose of the secondary network is to ensure fast and energy-efficient retransmission of only a small fraction of flits. Additionally, it needs to be simple and scalable with the number of nodes. These characteristics suggest a different topology than the primary network, which carries high traffic and serves general-purpose demands.

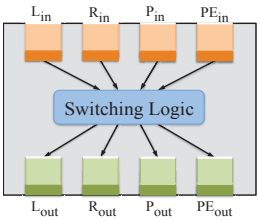
Therefore, while the primary network is a mesh, we choose a modified fat-tree network for the secondary network. There are three reasons for this. First, theoretical studies have established a fat-tree's benefits in terms of small average distance and scalability [31]. Secondly, there have been studies that show how to lay out a fat H-tree into VLSI structures efficiently [32], [33], [34], [35]. Thirdly, we substantially simplify the routing logic, and make the tree's layout modular by organizing it as a fat AVL tree [36]. An AVL tree has processors at every level of the tree — unlike a conventional tree, which has processors only at the leaves.

An AVL tree network with N nodes has a total number of links equal to N-1. Its diameter and the average distance grow only as logN. This enables the design of the secondary network to scale to large chips, while maintaining a small diameter and average distance. Moreover, the layout is modular because both tree leaves and intermediate tree nodes have processors. Finally, as we see next, we have developed a routing logic for the messages in the tree that is very simple.

Figure 5a shows a 16-node AVL tree with labeled nodes. The numbering is such that, for a node labeled i, all the nodes in its right subtree have numbers larger than i, and all nodes in its left subtree have numbers smaller than i.



(a) Fat AVL tree of 16 nodes.



(b) Tree switch.

Figure 5: An AVL tree and the internals of a tree switch.

Figure 5b shows the switch that we have developed for the AVL tree. The links flowing inward and outward from the switch are as follows: PE_{in} and PE_{out} connect to the local processing element (PE); L_{in} and L_{out} connect to the left child (L); R_{in} and R_{out} connect to the right child (R); and P_{in} and P_{out} connect to the parent node (P). Each of these links has a queue.

Figure 6 shows how the simple routing logic works. Each node has three registers, which contain: the number assigned to the node (*Self*), the largest number in the right subtree (R_{max}), and the smallest number in the left subtree (L_{min}). Given a flit in one of the input buffers, we compare its destination number (*Dest*) to these three numbers. *Dest* can fall into one of the five regions shown in Figure 6: lower than L_{min}, equal or higher than L_{min} but lower than *Self*, equal to *Self*, higher than *Self* but lower or equal to R_{max},

and higher than R_{max}. Based on where *Dest* falls in these five regions, the flit is routed to the parent, the left subtree, the local node, the right subtree, and the parent, respectively.

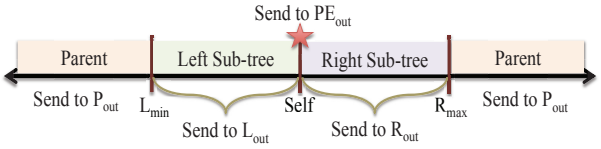


Figure 6: Simple routing logic for the tree switches.

The secondary network is very lightly used. Hence, we set the bandwidth of the links at the leaves to be 1/8th of that of the primary network links. To support the increasing utilization of the links at the upper tree levels, we double the width of the links every two levels.

V. Design Issues

A. Error Handling

To detect data flit errors, Sthira uses an 8-bit WCDMA-8 CRC code [37]. Whenever a flit is corrupted by a router, it fails the CRC at the next router and is dropped. Alternatives such as SECDED are not suitable for our use because of the possibility of multi-bit errors at low V_{dd}. Also, the area and energy overheads of a multi-bit ECC such as Orthogonal Latin Square Codes go against our primary goal of energy efficiency, considering the small number of errors that we see in our design. Hence, we use CRC.

Each router has one error counter designated for each of its neighbors. Whenever a flit from a neighbor is found faulty at a router, the corresponding counter is incremented. There are also counters that measure the total number of flits transmitted through each router. At the end of each epoch, the controller reads all these counters using light-weight control connections (in Sthira) or the secondary network (in Sthira+), and makes a decision on the new V_{dd} to apply.

The source node identifies an error condition using time-outs. A time-out occurs if the source does not receive an acknowledgment (ack) within a certain period. Sthira leverages the acks of transactions at the protocol layer (e.g., cache coherence transactions in shared-memory systems [38]). If the protocol does not have an ack for a given transaction, Sthira adds it. On a time-out, the source assumes the packet is dropped and retransmits the packet on the primary network (Sthira) or secondary one (Sthira+).

In scenarios of network congestion, even though long-duration message stalls are unlikely in NoCs [39], the source might pessimistically assume that a packet is dropped even though its flits are progressing slowly through the network. In this case, it re-sends the packet and the destination receives two copies of the same packet. By maintaining a pair-wise sequence number of the last acknowledged packet, duplicates are identified and dropped at the destination.

An error in control flits may result in misrouting or routing cycles. We implement the detection techniques detailed in [40] to comprehensively identify all such conditions (e.g., control path and VC bit errors) and drop corrupted flits, so that the source times-out and retransmits. We include all their overheads in our evaluation.

B. Multiple Routers per V_{dd} Domain

With multiple routers per V_{dd} domain, the controller operates in two steps. First, the controller chooses a V_{dd} for each router in the domain after measuring the error rate in the router. Second, the controller sets the V_{dd} of the domain to be the maximum of all the chosen V_{dd}s of the routers in that domain.

Due to this coarse granularity of V_{dd} control, the energy saved will be lower than with single-router V_{dd} domains. In reality, due to systematic variations, physically close routers are similarly affected by process variations, and their operational V_{dd} will likely be close.

C. Controller Characteristics and Aging

As shown in Figure 4, the PID controller is composed of lookup tables and adders. Each table has 128 entries, and each entry is 15 bits. Consequently, the controller's area is very modest. It is a simple circuit, with three adders and one table lookup in its longest path. Moreover, its operation is not time-critical. For example, for a 64-domain network, the controller only needs to operate 64 times every epoch (e.g., every 50 μs).

Over many months of NoC use, the routers may age and change their speed characteristics slightly. At that point, we may want to re-profile the routers, recompute the K_P, K_I and K_D controller constants, and reprogram the controller. Since aging effects occur at year-level timescale [41], reprogramming on a yearly basis adds negligible overhead.

D. Costs of the Sthira and Sthira+ Designs

Compared to a standard NoC, a Sthira/Sthira+ system has four main costs. Three of them are similar to a state-of-the-art heuristic approach called Tangle [11]; one cost is new. The three similar costs are:

1. Cost of voltage regulation: The multiple V_{dd} domains of the NoC are generated by voltage regulators (VRs). As discussed in Section II-C, VRs induce a power loss, but there are many different designs and the technology is evolving. Using data from the recent IBM POWER8 demonstration [22], we assume that providing multiple V_{dd} domains in the NoC wastes 10% of the NoC power.

2. Cost of error detection and counting: Our results in the evaluation include the energy and performance overheads of using link-level CRC and error counters in each router. The performance overhead of using CRC is negligible: it adds only one extra cycle for encoding at the source node, while the checking is done in parallel with flit processing at each of the routers in the flit's path.

3. Cost of retransmission buffers: The retransmission buffers that temporarily store transmitted packets at the source to enable retransmission in case of a fault, consume area and power. They may also cause a stall if they become full. Our results include the impact of these buffers.

The new cost is the formal controller system. In addition, for Sthira+, another cost is the secondary network, which replaces Sthira's control links between controller and routers. The formal controller and secondary network parameters are listed in Section VI. These two components introduce small overheads. We estimate that, on average, the controller introduces less than a 1% energy overhead to

Table I: Architecture and variation parameters. For the memory hierarchy, we give round-trip latencies from the core.

Core Parameters	
Frequency	1 GHz
Fetch, issue, and commit	2 per cycle
ROB; Ld/St queue	64 entries; 16/16 entries
Issue queue; I-fetch queue	64 entries; 32 entries
Branch (BR) predictor	Tournament (bimodal + 2-level)
BTB; history table	1024 entries, 2-way; 2048 entries
Memory System Parameters	
L1 data cache	32KB, 2-way, 2 cyc. latency, 64B line
L1 instr. cache	32KB, 2-way, 2 cyc. latency, 64B line
L2 cache	32MB shared, 64-bank static addr.
L2 cache bank	8-way, 64B line, 6 cyc. latency (local)
Main memory	260 cyc. latency, 4 mem. controllers
Main Network-on-Chip Parameters	
Topology; routing	8x8 2D-mesh; X-Y routing, wormhole
Num. virtual channels	4 per physical channel
Buffer depth per virtual channel	4
Min. V_{dd} tuning step	$10mV$, 20 cycles latency
Retransmission buffer depth	8
Link width	128b
Nominal V_{dd}	$825mV$ (10% guardband)
Length of an epoch	50,000 cycles (min.)
Num. routers in V_{dd} domain	From 1 to 64; 1 is default
Penalty to Sthira/Sthira+ NoCs	10% power due to V_{dd} regulation
Secondary Network Parameters	
Topology	AVL Tree
Routing	Least Common Ancestor routing
Link width	16b (lowest level), double every 2 levels
Number of buffers	One per link
Buffer depth	1
Nominal V_{dd}	$825mV$
Process Variation Parameters	
Tech. node; V_{dd} guardband	$11nm$; 10%
Total V_{th} (σ/μ)	12.5% (equal random & systematic)
Total L_{eff} (σ/μ)	6.25% (equal random & systematic)
Correlation range ϕ	0.1 (for both V_{th} and L_{eff})

the NoC (most of it static), while the secondary network introduces a 1.7% energy overhead (with comparable static and dynamic components). Our results in the evaluation include the energy and performance overheads introduced by these two components.

VI. Experimental Methodology

To evaluate the energy and performance of Sthira/Sthira+, we use several tools. First, we take the Verilog implementation of a wormhole-switched virtual channel router from [42], and develop a 3-stage look-ahead router similar to that proposed in [43]. Using the Synopsys Design Compiler, we perform timing analysis on this design to extract the 32 slowest paths and their netlists, for each stage. We use a 3-stage router instead of a more aggressive single-cycle one for two reasons. First, single-cycle routers tend to operate at slower clock rates. Second, they often rely on speculative operation, which makes them less energy efficient [44]. As we explore a futuristic chip design, we perform process variation modeling at $11nm$ using VARIUS-NTV [15]. The baseline chip comprises 64 nodes with private L1 caches, 64 routers with virtual channels, and 64 banks of a shared L2. To obtain the timing failure rates due to process variation, we first generate the chip floorplan. Next, the logical efforts extracted from the synthesis phase are used to enable a better delay modeling of the different stages of routers.

To obtain performance and energy consumption metrics, we use a cycle-level microarchitectural simulator of a multiprocessor chip with an NoC [40], [45]. Table I shows the architecture parameters. The baseline design of the main network has 64 routers in an 8x8 2D-mesh. Each router is associated with a core. A router has 5 physical channels (PCs), including the port from the local core to the router. There are 4 virtual channels (VCs) per PC — which are enough to avoid deadlocks. Each VC has a buffer depth of 4 flits. We use a wormhole-switched 2D-mesh network with deterministic X-Y routing (guaranteeing deadlock freedom) that uses credit-based flow control. The secondary network is a fat AVL tree. The router is implemented in structural RTL Verilog and synthesized using the Synopsys Design Compiler with Nangate $45nm$ open cell library. Based on rules given in [46] and technology parameters from the ITRS report [47], we scale it down to $11nm$.

The simulator also models the cores and caches. The cores are 2-issue out-of-order Alpha DEC EV4-like cores. Their frequency is 1GHz to save power and enable operation under a typical power envelope. Similar low frequencies have been used by researchers for large CMPs [40], [48].

We explore network configurations with 1 to 64 routers per V_{dd} domain. In the evaluation, the default system has 1 router per V_{dd} domain. We also vary the network size from 4x4 to 10x10 nodes. For each network, we use VARIUS-NTV to generate chips with representative variation profiles.

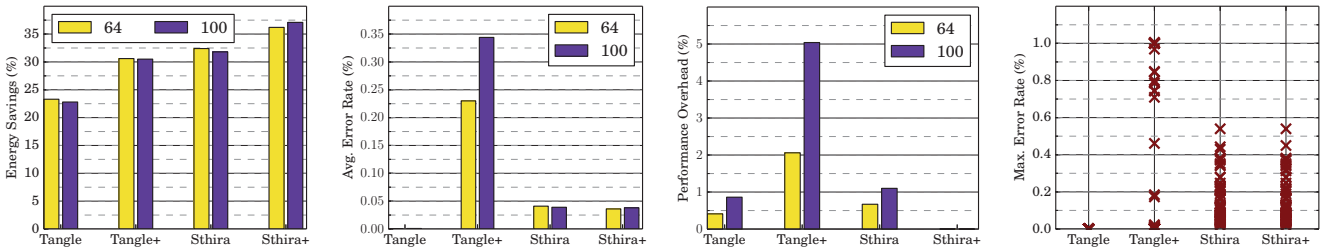
To estimate the static and dynamic power in the router buffers, crossbar, and clock distribution, we use our synthesis results from the Synopsys Design Compiler. The results agree with those in DSENT [49] generated by SPICE. For example, our synthesis experiments estimate $6.54mW$ for the power in the router buffers; DSENT reports $6.93mW$.

We use MATLAB to obtain the linearized error probability as a function of router V_{dd}, and design the controller. Using pole-zero analysis, we find the values for K_P, K_I, K_D that result in a stable controller with acceptable overshoots and undershoots (Section IV-A2).

For our experiments, we run 10 multi-programmed workloads. They are a subset of those used in [39]. Of these workloads, 3 are commercial (sap, sjas, and tpcw), 4 are engineering (429.mcf, 437.leslie3d, 459.GemsFDTD, and 483.xalancbmk) and 3 are scientific (art, ocean, and swim). We use SimPoint [50] to select representative windows of instructions for simulation, and we simulate about 10M instructions per thread after warm-up. In the experiments, we use a source time-out threshold of three times the round-trip latency, a target error rate of 0.05%, and an epoch duration of 50,000 processor cycles.

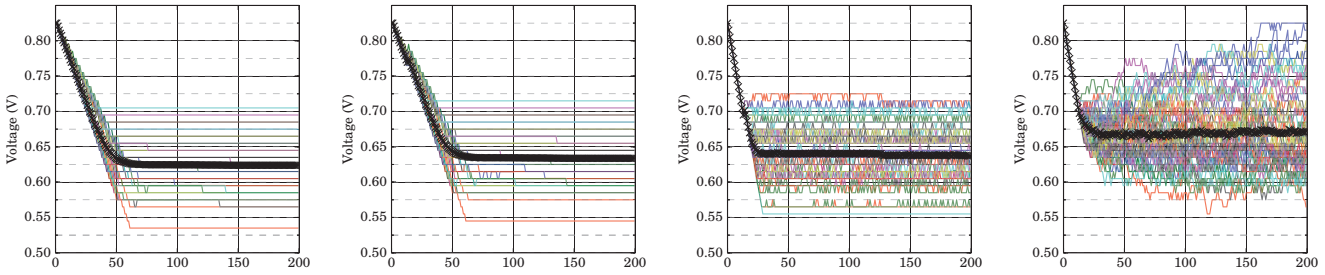
Our evaluation compares the energy consumption, performance, and error rates of the five NoC architectures of Table II. These architectures include, in addition to Sthira and Sthira+: (1) a nominal-V_{dd} plain NoC with a single V_{dd} domain (Baseline); (2) Tangle [11]; and (3) an enhanced version of Tangle where, on an error, only the erroneous router changes V_{dd}, not all routers in the flit path as Tangle does. This version is called Tangle+, and is an ad-hoc scheme that is more aggressive than Tangle and closer to Sthira.

In Tangle and Tangle+, we use a fixed V_{dd} tuning step of $10mV$. While [11] uses a variable step size, it requires



(a) Energy savings. (b) Average error rate. (c) Performance overhead. (d) Maximum error rate.

Figure 7: Comparing the energy savings, average error rate, performance overhead, and maximum error rate of the different architectures. Energy savings and performance overhead are normalized to the NoC without V_{dd} reduction (Baseline). The figure shows data for 64- and 100-router NoCs.



(a) Sthira+. (b) Sthira. (c) Tangle+. (d) Tangle.

Figure 8: Variation in the V_{dd} of each router over time (in epochs) in a 64-router NoC running the GemsFDTD application. Note that both voltage regulators in the hierarchical VR contribute to the voltage reduction.

Table II: NoC architectures compared.

Name	Architecture
Baseline	Nominal-V_{dd} plain NoC with a single V_{dd} domain.
Tangle	Tangle as described in [11].
Tangle+	Tangle except that, on an error, only the erroneous router changes V_{dd}, not all routers in the flit path.
Sthira	Proposed Sthira architecture.
Sthira+	Proposed Sthira architecture plus the secondary network.

prior knowledge of a "convergence voltage" $V_{AvgTest}$. We consider that such V_{dd} is not known for a variation-affected system. In any case, this change has minimal impact because it only affects the initial V_{dd} changes.

VII. Evaluation

A. Comparing the Different Schemes

We start the evaluation by comparing the schemes in Table II. Figure 7 compares them under four metrics: energy savings (a), average error rate (b), performance overhead (c), and maximum error rate (d). The figure shows data for 64- and 100-router NoCs. Next, we consider each metric in turn.

1) Comparing Energy Savings: Figure 7(a) shows the NoC energy savings relative to an NoC without V_{dd} reduction (Baseline). The data corresponds to the average of all the applications. We see that Sthira+ saves 36-37% of the NoC energy, while Sthira saves 32%. Tangle+ saves about 30% of the energy, and Tangle only saves 23%.

Sthira+ saves the most because it uses its advanced hardware (including the secondary network) to control V_{dd} well, and keep it the lowest. Sthira also has a formal controller, which helps it reduce the V_{dd}. However, on an error, the message is retransmitted on the same network. Since the V_{dd} of the routers is not changed until the beginning of the next epoch, the retransmitted message may suffer the same error again. As a result, Sthira will converge to a slightly higher V_{dd} — hence, saving less energy.

Tangle follows a different approach. As soon as a router detects an error, Tangle increases the V_{dd} without waiting for the epoch to complete. Moreover, Tangle increases the V_{dd} of all the routers in the path of the flit. As a result, the error rate will be low. However, the lack of formal control results in V_{dd} values that are higher than the other schemes. Consequently, Tangle saves the least energy.

Tangle+ follows the same eager approach as Tangle to increase the V_{dd} of a router immediately when an error is detected. However, it only increases the V_{dd} of that router. Hence, the energy savings are higher than in Tangle.

The impact of the different schemes on V_{dd} is shown in Figure 8. The figure has one plot per scheme, which shows the variation of the V_{dd} of each router over time (in epochs) in a 64-router NoC running the GemsFDTD application. Each plot also shows the average V_{dd} as a thicker line.

From the plots, we see that the steady-state average V_{dd} goes up from Sthira+ to Sthira, Tangle+, and Tangle. This change broadly explains the difference in energy savings in Figure 7(a). In addition, the plots show the very different convergence behavior of Sthira+ and Sthira on the one hand, and of Tangle+ and Tangle on the other. Specifically, in Sthira+ and Sthira, the V_{dd} of each router converges smoothly. This is thanks to the formal controller. However, in Tangle+ and Tangle, the V_{dd} of each router keeps oscillating. This produces suboptimal V_{dd} values — sometimes too high and therefore wasteful, and sometimes too low and so error-inducing.

2) Comparing the Average Error Rate: Figure 7(b) shows the average error rate across all the epochs, routers, and applications. There are bars for 64- and 100-router NoCs. Recall that we can specify a non-zero target error rate to the Sthira designs. We set it to 0.05% in our evaluation. As expected, we see that Sthira+ and Sthira have an average error rate close to the target one.

Tangle has a very low error rate. The reason is that Tangle is a very conservative scheme. Indeed, as soon as a router detects an error, the V_{dd} of all the routers in the flit path is immediately increased. On the other hand, Tangle+ suffers a much higher average error rate: 0.23% and 0.34% for 64- and 100-router NoCs, respectively. Tangle+ updates the router voltages in a hasty, ad hoc manner, reducing them to values lower than the minumum necessary for error-free operation. This leads to sharp increases in the error rate. Thus, due to the ad-hoc nature of Tangle+, its average error rate is not guaranteed to be below any particular value, and can be quite high.

3) Comparing Performance Overhead: Figure 7(c) shows the average overhead of program execution relative to execution on a machine with the Baseline NoC. There are bars for 64- and 100-router NoCs. We can see that Sthira+ has no visible performance overhead. It helps that, on an error, the packet is retransmitted on the unloaded secondary network.

Sthira has a performance overhead of 1% or less. The overhead is due to the retransmission over the main network of packets following errors. These packets add network congestion and more delay to the sender.

Tangle is a very conservative scheme that suffers very few errors (Figure 7(b)). As a result, it only has a small overhead. The main reason for the overhead is that routers are unavailable during V_{dd} changes.

Finally, Tangle+ has a 2-5% performance overhead. This high overhead results from the recovery from its relatively frequent errors (Figure 7(b)). These errors were the result of its aggressive yet ad-hoc nature. Consequently, Tangle+ is not attractive.

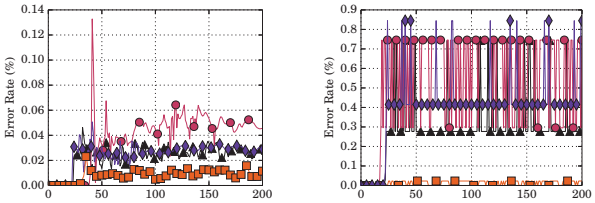
4) Comparing the Maximum Error Rate: Finally, Figure 7(d) shows the maximum error rate observed by each of the routers in a 64-router NoC. For each scheme, we have 64 data points. The data corresponds to running the GemsFDTD application.

For Sthira+ and Sthira, these data points correspond to the first error overshoot. After that, the error rate decreases and very soon converges to the target value. As seen in the figure, Sthira+ and Sthira have at least one router that reaches a 0.55% error rate.

We designed the Sthira controller for a maximum error rate of 0.1%. However, Figure 7(d) shows that some routers reach 0.55%. This is because the controller is designed for the average conditions of the 64 routers. Due to variation, routers deviate from average conditions and, therefore, some see higher error rates in their first overshoot. To ensure that we are always below the 0.1% error rate, we can provide a per-router controller. Alternatively, we can reduce the size of the minimum V_{dd} change step, currently set to a conservative 10 mV. We try the latter in Section VII-B3.

As shown in Figure 7(d), Tangle has a very small maximum error rate, but the opposite is true for Tangle+. Tangle+ has routers that reach 1% error rate. Moreover, it can be shown that the whole program execution in Tangle+ sees error-rate peaks of about 1%. Tangle+'s ad-hoc nature means that there is no error-rate convergence.

To see this effect, consider Figure 9, which shows how the error rate varies over time (in epochs) for Sthira (Chart (a)) and Tangle+ (Chart (b)). The figure corresponds to the execution of the GemsFDTD application. It shows the error rate of four routers: one with few errors, two with a medium number of errors, and one with many errors. Note that the Y axes of the two plots use numbers that are one order of magnitude apart.

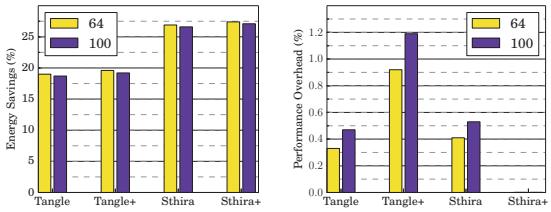


(a) Sthira. (b) Tangle+.

Figure 9: Variation of error rate over time (in epochs).

Figure 9(b) shows that the routers in Tangle+ keep the same error profile as time goes on. They do not converge. Figure 9(a), however, shows that the routers in Sthira have an initially high error rate and then converge to lower values.

5) Larger V_{dd} Domains: Figure 10 repeats the energy and performance plots of Figure 7 for NoCs with V_{dd} domains that cover 8 routers each. We can see that the energy reductions are smaller than with per-router V_{dd} domains. The reason is that the larger domains become less homogeneous in their variation parameters, and we still need to set a single V_{dd} for the whole domain. Overall, Sthira+ and Sthira save about 27% of the energy. This amount is still substantial, and about 9 percentage points higher than Tangle. In addition, their performance overhead is negligible. On the other hand, Tangle+ does not save much energy beyond Tangle and its performance overhead is about 1%.



(a) Energy savings. (b) Performance overhead.

Figure 10: Comparing the energy savings and performance overhead of the schemes with 8-router V_{dd} domains.

6) Summary: Overall, Sthira+ has the best scores in all dimensions. It has smooth V_{dd} convergence, no performance overhead and, compared to the only previously-proposed scheme (Tangle), reduces the NoC energy by an additional 13% for single-router V_{dd} domains (and by 9% for 8-router domains). Its effectiveness is in part due to the secondary network, which highly-efficiently retransmits the packets that failed.

A cheaper and, to our opinion, more cost-effective alternative is Sthira, which does not have the secondary network. Sthira has the same smooth convergence of router V_{dd} and

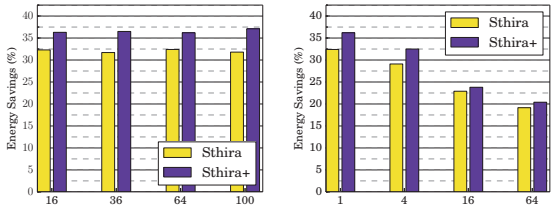
error rates as Sthira+, thanks to a formal controller. Its energy savings and performace overhead are slightly less favorable. Still, compared to Tangle, Sthira has the same minimal performance overhead and reduces the NoC energy by an additional 9% for single-router V_{dd} domains (and by 9% for 8-router domains).

Tangle+ is not attractive. While, for single-router domains, it saves nearly as much energy as Sthira, it slows down execution by 2-5%. In addition, even in steady state, it suffers large oscillations in error rate and, to a lesser degree, V_{dd}. Further, it provides no guarantees on error-rate limits.

B. Sthira/Sthira+ Characterization

We now examine some aspects of Sthira and Sthira+.

1) Energy Savings for Different NoC Sizes: The careful design of the controller in Sthira and Sthira+ enables these architectures to retain the energy savings (about 32% and 37%, respectively) across NoC sizes, as shown in Figure 11a. Due to its advanced design, Sthira+ is able to save more energy than Sthira for all network sizes.



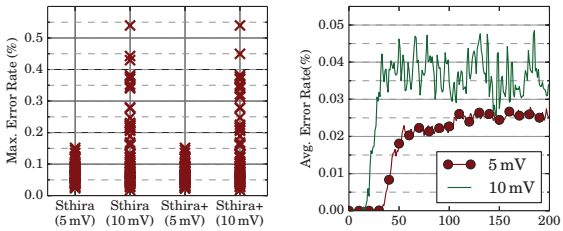
(a) Across NoC sizes. (b) Across V_{dd} domain sizes.

Figure 11: Energy Savings across NoC sizes and V_{dd} domain sizes in Sthira and Sthira+

2) Energy Savings for Different V_{dd} Domain Sizes: The energy savings is expected to reduce with increasing size of the V_{dd} domains in the chip. With larger V_{dd} domains, the V_{dd} of an entire domain is set conservatively to the maximum of the V_{dd} of all the routers in the domain. Hence, the savings decrease as shown in Figure 11b. From this figure, it can also be seen that, even in the case of a single V_{dd} domain for the entire chip (i.e., size of 64), Sthira and Sthira+ are able to save 19-20% of the NoC energy.

3) Reducing the Minimum V_{dd} Step Size: The output of the controller in Sthira and Sthira+ is limited by the minimum V_{dd} step that it is allowed to generate. As the minimum V_{dd} step becomes smaller, two observable metrics improve. First, the maximum error rate (the peak overshoot) is lower because the controller can more finely adjust the V_{dd}. Second, the variation of error rate with time is smoother, nearly eliminating the majority of sharp transients. The first result is plotted in Figure 12a, which shows the maximum error rate coming down to 0.16% from 0.55% when a minimum step of $5mV$ is used. The second improvement is shown in Figure 12b, which demonstrates the difference in the behavior of average error rate with time, for $5mV$ and $10mV$ steps. The plot corresponds to Sthira. Smaller steps are expected to be available in the future.

4) Secondary Network Utilization in Sthira+: Sthira+ includes an additional network in its design to achieve higher energy savings and further reduce the performance overheads of Sthira. The novel design of this network along with the tight control over error rates keeps the utilization



(a) Max Error Rate. (b) Error rate vs. time.

Figure 12: Impact of the minimum V_{dd} step on Sthira and Sthira+. Chart (b) corresponds to Sthira.

low. Figure 13 shows the average utilization of the root and leaf links in the secondary network, for different NoC sizes ranging from 16 to 100 routers. For a given setting, the two utilizations are overlaid to save space. We see that, even for the 100-router NoC, the utilization of the root links does not go above 10% for any application. For smaller NoC sizes, the utilization is much smaller.

C. Design Space Exploration

We study some design parameters in Sthira and Sthira+.

1) Controller Activation Threshold: The controller is activated for the first time when the V_{dd} decreases to the point where the error rate of a router reaches a certain *activation error threshold* — i.e., the router reaches the modeled region (Section IV-A1). We experiment with different activation error thresholds. In all cases, we keep the same target error rate. Very low activation error thresholds increase the time to converge due to the nonlinearity near the lower knee of the error rate curve. On the other hand, a large activation error threshold activates the controller too late, resulting in higher errors and also long convergence times. It can be shown that an activation error threshold of 0.001% keeps the convergence times and error rates acceptable.

2) Reference Error Rate: The reference or target error rate ($E_\circ$) determines the tradeoff between resilience and energy savings. A very small $E_\circ$ may not reduce V_{dd} guardbands much; a large $E_\circ$ may yield diminishing benefits, due to both the many message retransmissions required and the fact that the error rate increases exponentially with V_{dd} reduction. Our default $E_\circ$ is 0.05%. We find that, if we increase $E_\circ$ further, the steady state V_{dd}s are only slightly lower (by at most 1%), and the added retransmission overheads dampen any benefits. On the other hand, if we set $E_\circ$ to 0.01%, we lose about 3% of energy savings. In all cases, Sthira is able to keep the average error rate close to the chosen $E_\circ$. Indeed, Figure 14 shows the measured average error rate across epochs for different reference error rates.

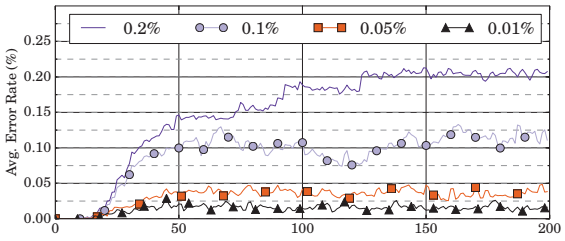


Figure 14: Avg. error rate for different reference error rates.

3) Link Width of the Secondary Network: For the secondary network, reducing the link bandwidth saves static power while increasing utilization. We find that the average

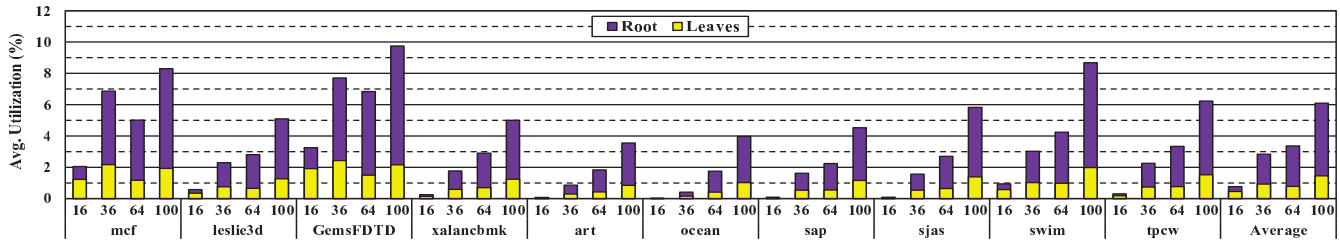


Figure 13: Average utilization of the root and leaf links in the secondary network for different NoC sizes. For a given setting, the two utilizations are overlaid to save space.

utilization of the secondary network at the root node grows only slowly, and stays below 10% for link widths no lower than the ones we use as baseline. However, if we further reduce the link widths by half, the average utilization at the root links can go up to 26% for some applications.

4) Fat-Tree Bandwidth: To reduce congestion in a fat tree, the bandwidth of the links is gradually increased as the links get closer to the root level. In Sthira+, maintaining a constant link bandwidth causes the average utilization at the root level to reach 35%, resulting in a noticeable performance overhead. Doubling the bandwidth at every level brings down the utilization to less than 1%, but increases area and energy. Hence, we choose to double the bandwidth every other layer. This balances the energy-utilization tradeoff.

D. Discussion: Other Sources of Variation

This paper focused on reducing V_{dd} guardbands added to address process variation and, potentially, circuit wearout. However, the control-theoretic design of the controller can potentially allow it to adapt to more dynamic changes — e.g., temperature changes due to workload variations. In such scenarios, a controller could be designed to dynamically learn and adapt V_{dd} to these changes. A detailed analysis of such scenarios is beyond the scope of this paper.

VIII. Related Work

The impact of process variation on NoCs has been studied by Nicopoulos et al. using rigorous circuit analysis [51]. Their analysis does not include a fault model or the impact of faults at the system level. In [7], Li et al. show that a high degree of process variation can force major design modifications to the underlying network architecture. Ogras et al. [52] explore the effectiveness of multiple V_{dd}-frequency domains in NoCs when dealing with a deep sub-micron process. Lefurgy et al. [16] use a feedback regulator to reduce the V_{dd} guardbands by continuously monitoring the available timing margin. Such approach requires continuous pipeline monitoring, and responds immediately on an event.

Some proposals compensate for the timing variations of links by time borrowing or cycle tuning [53], [54]. Time borrowing/stealing is a technique that has been used to tackle process variations in the processor pipeline [55], [56], [57]. Such techniques are mostly applied statically, at manufacturing time and require circuit-level modification to the underlying router design. This limits these techniques in adapting to the runtime conditions of the system.

Many techniques have been proposed to improve the fault tolerance of both links and routers in the presence of permanent and transient faults [40], [58], [59]. These solutions may incur higher overheads as they are proactive, and tackle all types of faults in the same way. Our goal is to adjust the circuit parameters to address only timing failures.

Pothukuchi et al. [23] advocate the general use of control-theoretic methodologies in controllers to meet multiple objectives in processors. The application of control theory to save power is explored by Wu et al. [30] in multiple clock domain processors, and by Raghavendra et al. [60] in data centers. We have used some of their insights. However, we focus on a different domain, namely NoC routers. Also, unlike [30], [60], we do not assume multiple frequency domains, since they necessitate the use of asynchronous queues to communicate across domains.

Chen et al. [61] use a PI controller to set the uncore DVFS. They measure the memory access time as a function of frequency for some programs and generate a PI controller. We use control theory for a different goal: to reduce wasteful V_{dd} guardbands arising from process variations. Consequently, our approach and design are different.

IX. Conclusion

This paper presented Sthira, a scheme that dynamically minimizes the V_{dd} of groups of routers in a variation-prone NoC at constant frequency using formal control-theory methods. We also enhanced Sthira with a low-cost secondary network that retransmits faulty packets for higher energy efficiency. The enhanced scheme is called Sthira+.

We evaluated Sthira and Sthira+ with simulations of NoCs with 64–100 routers, and compared them to state-of-the-art designs. In an NoC with 8 routers per V_{dd} domain, our schemes reduced the average energy consumption of a conventional NoC by 27%; in a futuristic NoC with one router per V_{dd} domain, Sthira+ and Sthira reduced the average energy consumption of a conventional NoC by 36% and 32%, respectively. The performance impact was negligible. These savings are obtained after taking into account the power losses in voltage regulators, and are 9–13 percentage points higher than in the best existing ad hoc proposals. Moreover, we showed that formal control is essential to provide guarantees on the convergence, stability, and maximum resulting error rates. Finally, while the secondary network helped Sthira+ attain higher energy savings, it has a non-negligible hardware cost. Hence, Sthira is the most cost-effective design.

Acknowledgments

This work was supported in part by NSF under grants CCF-1536795 and CCF-1649432.

REFERENCES

[1] S. Y. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, 2005.

[2] ——, "Future of Interconnect Fabric: A Contrarian View," in *SLIP*, 2010.

[3] N. Carter, A. Agrawal, S. Borkar, R. Cledat, H. David, D. Dunning, J. Fryman, I. Ganev, R. Golliver, R. Knauerhase, R. Lethin, B. Meister, A. Mishra, W. Pinfold, J. Teller, J. Torrellas, N. Vasilache, G. Venkatesh, and J. Xu, "Runnemede: An Architecture for Ubiquitous High-Performance Computing," in *HPCA*, Feb. 2013.

[4] S. Dighe, S. Vangal, P. Aseron, S. Kumar, T. Jacob, K. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V. De, and S. Borkar, "Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 184–193, 2011.

[5] X. Fu, T. Li, and J. A. B. Fortes, "Architecting Reliable Multi-core Network-on-chip for Small Scale Processing Technology," in *DSN*, 2010.

[6] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart, "A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 173 –183, 2011.

[7] B. Li, L.-S. Peh, and P. Patra, "Impact of Process and Temperature Variations on Network-on-Chip Design Exploration," in *NOCS*, 2008.

[8] S. Hemmert, "From Petascale to Exascale: R & D Challenges for HPC Simulation Environments," Hardware Architecture White Paper, Mar. 2011.

[9] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Zeisler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," in *MICRO*, Dec. 2003.

[10] B. Greskamp, L. Wan, U. R. Karpuzcu, J. J. Cook, J. Torrellas, D. Chen, and C. B. Zilles, "Blueshift: Designing Processors for Timing Speculation from the Ground Up," in *HPCA*, 2009.

[11] A. Ansari, A. Mishra, J. Xu, and J. Torrellas, "Route-Oriented Dynamic Voltage Minimization for Variation-Afflicted, Energy-Efficient On-Chip Networks," in *HPCA*, 2014.

[12] A. Bacha and R. Teodorescu, "Dynamic Reduction of Voltage Margins by Leveraging On-chip ECC in Itanium II Processors," in *ISCA*, 2013.

[13] ——, "Using ECC Feedback to Guide Voltage Speculation in Low-Voltage Processors," in *MICRO*, 2014.

[14] X. Liang, R. Canal, G.-Y. Wei, and D. Brooks, "Replacing 6T SRAMs with 3T1D DRAMs in the L1 Data Cache to Combat Process Variability," *IEEE Micro*, vol. 28, no. 1, pp. 60–68, Jan. 2008.

[15] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torrellas, "VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages," in *DSN*, 2012.

[16] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, "Active Management of Timing Guardband to Save Energy in POWER7," in *MICRO*, 2011.

[17] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-L. Lu, "Reducing Cache Power with Low-Cost, Multi-bit Error-Correcting Codes," in *ISCA*, 2010.

[18] L. Chang, R. Montoye, B. Ji, A. Weger, K. Stawiasz, and R. Dennard, "A Fully-Integrated Switched-Capacitor 2:1 Voltage Converter with Regulation Capability and 90% Efficiency at 2.3A/mm2," in *VLSIC*, Jun. 2010.

[19] "Intel® Xeon® Processor E3-1200 v3 Product Family Datasheet," http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-e3-1200v3-vol-1-datasheet.pdf, Jul. 2014.

[20] H. R. Ghasemi, A. Sinkar, M. Schulte, and N. S. Kim, "Cost-Effective Power Delivery to Support Per-Core Voltage Domains for Power-Constrained Processors," in *DAC*, Jun. 2012.

[21] F. Ishihara, F. Sheikh, and B. Nikolic, "Level conversion for dual-supply systems," *IEEE Trans. VLSI Syst.*, vol. 12, no. 2, pp. 185–195, Feb. 2004.

[22] E. Fluhr, S. Baumgartner, D. Boerstler, J. Bulzacchelli, T. Diemoz, D. Dreps, G. English, J. Friedrich, A. Gattiker, T. Gloekler, C. Gonzalez, J. Hibbeler, K. Jenkins, Y. Kim, P. Muench, R. Nett, J. Paredes, J. Pille, D. Plass, P. Restle, R. Robertazzi, D. Shan, D. Siljenberg, M. Sperling, K. Stawiasz, G. Still, Z. Toprak-Deniz, J. Warnock, G. Wiedemeier, and V. Zyuban, "The 12-Core POWER8^{TM} Processor With 7.6 Tb/s IO Bandwidth, Integrated Voltage Regulation, and Resonant Clocking," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 10–23, Jan. 2015.

[23] R. P. Pothukuchi, A. Ansari, P. Voulgaris, and J. Torrellas, "Using Multiple Input, Multiple Output Formal Control to Maximize Resource Efficiency in Architectures," in *ISCA*, 2016.

[24] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

[25] E. W. Weisstein. "Erf." From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Erf.html

[26] ——. "Gompertz Curve." From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/GompertzCurve.html

[27] ——. "Logistic Distribution." From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/LogisticDistribution.html

[28] ——. "Sigmoid Function. From MathWorld – A Wolfram Web Resource". [Online]. Available: http://mathworld.wolfram.com/SigmoidFunction.html.

[29] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, *Control System Design*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[30] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors," in *ASPLOS*, 2004.

[31] C. E. Leiserson, "Fat-trees: Universal Networks for Hardware-efficient Supercomputing," *IEEE Trans. Comput.*, vol. 34, no. 10, pp. 892–901, Oct. 1985.

[32] D. Ludovici, F. Gilabert, S. Medardoni, C. Gomez, M. Gomez, P. Lopez, G. N. Gaydadjiev, and D. Bertozzi, "Assessing Fat-tree Topologies for Regular Network-on-chip Design under Nanoscale Technology Constraints," in *DATE*, 2009.

[33] H. Matsutani, M. Koibuchi, and H. Amano, "Performance, Cost, and Energy Evaluation of Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network," in *IPDPS*, Mar. 2007.

[34] H. Matsutani, M. Koibuchi, Y. Yamada, D. Hsu, and H. Amano, "Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 8, pp. 1126–1141, Aug. 2009.

[35] Z. Wang, J. Xu, X. Wu, Y. Ye, W. Zhang, M. Nikdast, X. Wang, and Z. Wang, "Floorplan Optimization of Fat-Tree Based Networks-on-Chip for Chip Multiprocessors," *IEEE Trans. Comput.*, vol. 99, p. 1, 2012.

[36] G. M. Adel'son-Vel'sky and Y. M. Landis, "An Algorithm for the Organization of Information," *Doklady Akademii Nauk*

USSR, vol. 146, no. 2, pp. 263–266, 1962.

[37] P. Koopman and T. Chakravarty, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks," in *DSN*, 2004.

[38] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy, "The Directory-based Cache Coherence Protocol for the DASH Multiprocessor," in *ISCA*, May 1990, pp. 148–159.

[39] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A Case for Heterogeneous On-chip Interconnects for CMPs," in *ISCA*, 2011.

[40] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring Fault-Tolerant Network-on-Chip Architectures," in *DSN*, 2006.

[41] K. Bhardwaj, K. Chakraborty, and S. Roy, "Towards Graceful Aging Degradation in NoCs Through an Adaptive Routing Algorithm," in *DAC*, Jun. 2012.

[42] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *HPCA*, 2001.

[43] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A Low Latency Router Supporting Adaptivity for On-chip Interconnects," in *DAC*, Jun. 2005.

[44] A. Kumar, P. Kundu, A. P. Singh, L.-S. Peh, and N. K. Jha, "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS," in *ICCD*, 2007.

[45] R. Das, O. Mutlu, T. Moscibroda, and C. Das, "Application-aware Prioritization Mechanisms for On-chip Networks," in *MICRO*, Dec. 2009.

[46] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, Jul. 1999.

[47] "International Technology Roadmap for Semiconductors (ITRS)," 2014 Update.

[48] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. Kumar, R. Ramanarayanan, V. Erraguntla, J. Howard, S. R. Vangal, S. Dighe, G. Ruhl, P. A. Aseron, H. Wilson, N. Borkar, V. De, and S. Borkar, "A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS," in *ISSCC*, 2012.

[49] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *NOCS*, 2012.

[50] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," in *ASPLOS*, 2002, pp. 45–57.

[51] 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 Trans. Dependable Secure Comput.*, vol. 7, no. 3, pp. 240–254, Jul. 2010.

[52] Ü. Y. Ogras, R. Marculescu, and D. Marculescu, "Variation-adaptive Feedback Control for Networks-on-chip with Multiple Clock Domains," in *DAC*, 2008.

[53] A. K. Mishra, R. Das, S. Eachempati, R. R. Iyer, N. Vijaykrishnan, and C. R. Das, "A Case for Dynamic Frequency Tuning in On-chip Networks," in *MICRO*, 2009.

[54] M. Simone, M. Lajolo, and D. Bertozzi, "Variation Tolerant NoC design by Means of Self-calibrating Links," in *DATE*, 2008.

[55] X. Liang and D. Brooks, "Mitigating the Impact of Process Variations on Processor Register Files and Execution Units," in *MICRO*, 2006.

[56] X. Liang, G.-Y. Wei, and D. Brooks, "Revival: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency," *IEEE Micro*, vol. 29, no. 1, pp. 127–138, Jan. 2009.

[57] A. Tiwari, S. R. Sarangi, and J. Torrellas, "ReCycle: Pipeline Adaptation to Tolerate Process Variation," in *ISCA*, 2007.

[58] D. Bertozzi, L. Benini, and G. de Micheli, "Low Power Error Resilient Encoding for On-Chip Data Buses," in *DATE*, 2002.

[59] T. Dumitras, S. Kerner, and R. Marculescu, "Towards On-chip Fault-tolerant Communication," in *ASP-DAC*, 2003.

[60] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center," in *ASPLOS*, 2008.

[61] X. Chen, Z. Xu, H. Kim, P. Gratz, J. Hu, M. Kishinevsky, and U. Ogras, "In-network Monitoring and Control Policy for DVFS of CMP Networks-on-Chip and Last Level Caches," in *NOCS*, May 2012.