Tangle: Route-Oriented Dynamic Voltage Minimization for Variation-Afflicted, Energy-Efficient On-Chip Networks *

Amin Ansari,† Asit Mishra,‡ Jianping Xu,‡ Josep Torrellas †
† University of Illinois at Urbana-Champaign
amina,torrella@illinois.edu
‡ Intel Corporation
asit.k.mishra,jianping.xu@intel.com

Abstract

On-chip networks are especially vulnerable to within-die parameter variations. Since they connect distant parts of the chip, they need to be designed to work under the most unfavorable parameter values in the chip. This results in energy-inefficient designs. To improve the energy efficiency of on-chip networks, this paper presents a novel approach that relies on monitoring the errors of messages as they traverse the network. Based on the observed errors of messages, the system dynamically decreases or increases the voltage (V_{dd}) of groups of network routers. With this approach, called Tangle*, the different V_{dd} values applied to different groups of network routers progressively converge to their lowest, variation-aware, error-free values — always keeping the network frequency unchanged. This saves substantial network energy. In a simulated 64-router network with 4 V_{dd} domains, Tangle reduces the network energy consumption by an average of 22% with negligible performance impact. In a future network design with one V_{dd} domain per router, Tangle lowers the network V_{dd} by an average of 21%, reducing the network energy consumption by an average of 28% with negligible performance impact.*

1. Introduction

Although successive CMOS generations continue to enable higher transistor integration, they have long deviated from the energy density predicted by classical scaling. As a result, energy and power consumption have emerged as the main constraints for current chip designs [35]. It is therefore crucial to devise techniques that substantially increase the energy efficiency of multi- and many-cores.

At the same time, as feature sizes shrink, parameter variations are gaining prominence. Variations are the deviation of process, temperature, and V_{dd} parameters from their nominal specifications [3]. As the first line of defense against parameter variations, designers increase guardbands, which make chips less energy efficient.

A component of multi- and many-core chips that is especially vulnerable to variations is the on-chip interconnection network. This is because the network connects distant parts of the chip which, due to variations, exhibit different speed and power characteristics. The network has to be designed conservatively, to work under the most unfavorable parameter values in the chip. This results in energy-inefficient designs. On-chip networks can already consume a substantial fraction of the on-chip power — potentially up to 30–40%, according to the literature [4, 6, 8, 13, 17, 29]. Conservative future network designs, needed to tolerate parameter variations, may be unable to reduce the value of this fraction much.

One of the most powerful knobs for energy-efficient design is V_{dd}. This is because V_{dd} has a strong impact on both dynamic and static energy. The V_{dd} guardbands present in the network to tolerate parameter variations offer an opportunity for energy savings: due to variations, the guardbands in some areas of the chip are likely to be significantly over-provisioned. The insight in our work is to reduce these guardbands to save energy, while being mindful not to reduce V_{dd} so much to cause timing errors in the network. Interestingly, well-known error detection and tolerance mechanisms in the network can be used to find out when V_{dd} has reached a tolerable lower bound.

Despite this observation, finding which groups of routers should lower their V_{dd} and by how much in a distributed environment is not trivial. Thus, this paper presents a mechanism, called *Tangle*, that dynamically measures the errors of messages and scales the V_{dd} of groups of routers to an error-free minimum. The frequency of the network remains unchanged.

Tangle augments a multi- or many-core chip that has multiple V_{dd} domains and the capability to perform dynamic V_{dd} scaling. Tangle monitors the errors of messages as they traverse the network. If errors are observed, Tangle dynamically increases the V_{dd} of the router groups used by the erroneous messages. Tangle also periodically decreases the V_{dd} of all the routers. With this approach, the V_{dd} values applied to different groups of routers progressively converge to their lowest, variation-aware, error-free values. This saves substantial network energy. Tangle has no noticeable performance overhead because it does not reduce frequencies and keeps the error rate to a bare minimum.

We evaluate Tangle with simulations of a variation-afflicted 64-router network. With 4 V_{dd} domains in the network, Tangle reduces the network energy consumption by an average of 22% with negligible performance impact. In a future network design with one V_{dd} domain per router, Tangle lowers the network V_{dd} by an average of 21%, reducing the network energy consumption by an average of 28% with negligible performance impact.

This paper is organized as follows. Section 2 presents a motivation and background; Sections 3 and 4 describe the Tangle architecture and some associated design aspects; Sections 5 and 6 evaluate Tangle; and Section 7 covers related work.

2. Motivation and Background

2.1. Process Variations & Trading Energy for Resilience

As transistor feature sizes continue to decrease, process variations have become a major concern for chip manufacturers [3].

*This work was supported in part by NSF under grant CCF-1012759 and a CCC Computing Innovation Fellowship; DARPA under UHPC Contract HR0011-10-3-0007 and PERFECT Contract HR0011-12-2-0019; and DOE ASCR under Award Numbers DE-FC02-10ER2599 and DE-SC0008717. Amin Ansari is now with Qualcomm Inc., San Diego, CA.

Variations manifest at many levels: wafer-to-wafer (W2W), die-to-die (D2D), and within-die (WID). In this paper, we focus on WID variations [31, 42], which are a challenge for network-on-chip (NoC) designs. WID variations include both random and systematic components. The former are primarily due to varying dopant concentrations; the latter are typically caused by the impreciseness of the manufacturing equipment. Systematic variations show a strong spatial correlation, which means that close-by devices have similar systematic values.

WID process variations result in chips that run at lower frequencies and consume more static power than variation-free ones. Indeed, while variation causes some paths to become slower and others faster, the slowest paths determine the chip frequency. Moreover, the exponential nature of the static power dependence on a transistor's threshold voltage (V_{th}) means that transistors that leak little save less power than the power consumed by those that leak a lot. Finally, given the resulting uncertainties, variation-afflicted designs include wider V_{dd} and frequency (f) guardbands, which make them even less efficient.

The wider guardbands present in variation-afflicted designs open up the opportunity to improve energy efficiency by carefully trading it off for resilience. The idea involves aggressively reducing the guardbands of the design while, at the same time, providing the capability to detect and correct any resulting errors. The advantage of this approach is that substantial energy savings can be attained.

We see this approach as part of a wider trend of trading-off energy efficiency for resilience. For example, some researchers propose timing speculation in the pipeline, such as in Razor [10] and BlueShift [15]. In these designs, a core is clocked at an unsafe f while a special latch or an extra core can detect and correct any resulting error. Other researchers propose to save energy by reducing the refresh rate of on-chip DRAM memories, while increasing the strength of their ECC codes [51]. Finally, other designs are made tolerant to approximate computations to save energy [11].

This paper is inspired by this trend. Our goal is to save energy, while always keeping the execution correct.

2.2. Fault Model

We model a variation-afflicted NoC at an aggressive process technology, as we progressively reduce its V_{dd} at a fixed f. We use the VARIUS-NTV [21] model to model process variations and the resulting timing faults that occur when the guardbands of the devices are reduced.

As the V_{dd} decreases, some logic paths that, due to process variation, are especially slow, start missing timing under certain logic values [42]. These are timing violations that can be categorized as intermittent faults. As the V_{dd} keeps decreasing, the fault rate increases. The model provides a certain probability of a timing fault for each path in the NoC. When a fault in the NoC occurs, we assume that it causes an error in the NoC that would result in an incorrect program execution. Consequently, we need to detect it and handle it.

2.3. Enabling Technology: Inexpensive V_{dd} Regulation

To adapt the NoC to WID process variations, we design it with multiple V_{dd} domains. In this way, the NoC regions whose systematic variations make them slow can be operated at relatively higher V_{dd}, while those that are fast are operated at relatively lower V_{dd}. However, using many switching voltage regulators (SVRs), either on- or off-chip, to support multiple V_{dd} domains is expensive [7]. First, their power efficiency is around 90% in the best case, and often lower under real operating conditions. Second, on-chip SVRs take substantial area, while off-chip SVRs require many pins and board hardware. For these reasons, with tried-and-tested current technology, it is hard to justify more than a handful of on-chip SVRs, even for a large chip — e.g., 4–8 SVRs.

However, upcoming technology will change this. Voltage regulators for a large chip may be hierarchical [14]. The first level is composed of one or a handful of SVRs, placed on a stacked die with devices optimized for the SVR inductances. The second level is composed of many on-chip low-drop-out (LDO) voltage regulators. Each LDO is connected to one of the first-level SVRs and provides the V_{dd} for, say, a single router. LDOs have a very high efficiency if the ratio of their output (V_O) to input (V_I) voltages is close to 1. Thanks to systematic variation, the LDOs in a region of the chip will need to provide very similar V_O. Since they take their V_I from the same first-level SVR and their V_O is similar, their efficiency can be close to 95%. In addition, their area is negligible: their hardware reuses a power-gating circuit. Such circuit is already likely to be present to power-gate the router. Finally, level converters between the resulting V_{dd} domains can be designed efficiently, by combining them with the latches [20].

Therefore, we propose two designs of Tangle. The first one applies to current chips and uses only a handful of V_{dd} domains in the NoC; the second applies to a future chip that uses one LDO (and hence one V_{dd} domain) per NoC router.

3. Tangle Architecture

In this section, we first present the high-level idea behind Tangle. Then, we describe the architecture in detail.

3.1. Key Idea

Given a large multi-core chip where the NoC has multiple V_{dd} domains, our goal is to operate the NoC at the minimum possible V_{dd} values for these domains, without sacrificing correctness or incurring any noticeable performance penalty. As discussed in Section 2, reducing V_{dd} can cause timing errors in the logic paths. Therefore, we are interested in an effective architectural mechanism that finds the lowest possible V_{dd} at which groups of routers can operate with very low error rates. For this, we will take advantage of inexpensive NoC mechanisms to detect and tolerate errors.

Tangle is our proposed low-cost solution for finding the V_{dd} of the NoC domains. Rather than using static V_{dd} values, Tangle determines these V_{dd} dynamically, adapting to the workload and temperature conditions. In this way, Tangle avoids exhaustive, expensive testing at manufacturing time [38]. Furthermore, dynamic V_{dd} setting also enables Tangle to adapt to aging effects [48, 50].

To guarantee the correct operation of the system, Tangle sets the initial V_{dd} of all the NoC routers to the nominal V_{dd} of the target process technology (e.g., $800mV$). This nominal V_{dd} guarantees fault-free operation. Then, Tangle gradually decreases the V_{dd} of groups of routers, and monitors the opera-

tion to maintain correctness. To monitor the system behavior, Tangle uses error detection codes, which can detect errors when critical paths in the circuit become too slow.

Error detection can happen at different levels. We can either perform end-to-end (network level) checking or switch-to-switch (link level) checking. In end-to-end checking, encoding happens at a message's source node and the check at the destination node. In contrast, in switch-to-switch checking, an error check is performed at every single router. The main advantage of end-to-end checking is the overall low cost of error checking, both in terms of energy and area [36]. This gives us the opportunity to use stronger error detection codes as well. On the other hand, switch-to-switch checking improves the error detection latency and reduces retransmission buffer size. Given that our overriding goal is energy efficiency, Tangle performs end-to-end error detection. For this purpose, we use the end-to-end error detection and retransmission scheme proposed in [23]. We account for all the scheme's overheads and do not extend it with new routing algorithms or flow control schemes. As shown in [23], for low error rates, which is certainly the case for our scheme, the performance overhead and buffer requirements are not drastically different between end-to-end and switch-to-switch error detection and retransmission techniques.

In Tangle, V_{dd} tuning is performed by a *Reliability Management Unit (RMU)*. The RMU manages the V_{dd} of the domains in the NoC. Whenever an error is detected at an end node, a signal is sent to the RMU, requesting an increase in the V_{dd} of the router groups in the path taken by the erroneous message. For this, we take advantage of deterministic routing. Given the source and destination of a flit, the RMU knows which routers this flit went through. Therefore, the error has happened in one of the routers in the path. After the V_{dd} increase, the flit is retransmitted from the source node.

In the next few sections, for simplicity and without loss of generality, we will assume that each NoC router has a V_{dd} domain of its own. Later, we describe the small changes required when each V_{dd} domain has a group of routers.

3.2. Supply Voltage Tuning

Tangle starts with a high V_{dd} value and gradually reduces it until it observes errors in the NoC. To achieve this, we first introduce the concept of *epoch*. An epoch is the fixed time interval between two consecutive reductions of the V_{dd} of all of the routers in the NoC. At the beginning of each epoch, Tangle reduces the V_{dd} of all the routers in the system by ΔV_{dec}. Then, for the rest of the epoch, Tangle monitors the errors that occur in the system. If it observes an error, it increases the V_{dd} for a subset of the routers by ΔV_{inc}, as will be described later.

An epoch needs to be large enough so that the performance overhead of V_{dd} tunings is minimal. At the same time, it should be short enough to allow routers to attain their target V_{dd} with a short delay. As will be discussed in the evaluation, we choose an epoch of 50,000 cycles, which allows the routers to reach their target V_{dd} after a few tens of epochs (i.e., in a few milliseconds).

Figure 1 shows an example of the changes in the V_{dd} of a router over time. The figure shows nine consecutive epochs. The initial V_{dd} is around $800mV$, and it gradually reduces. At

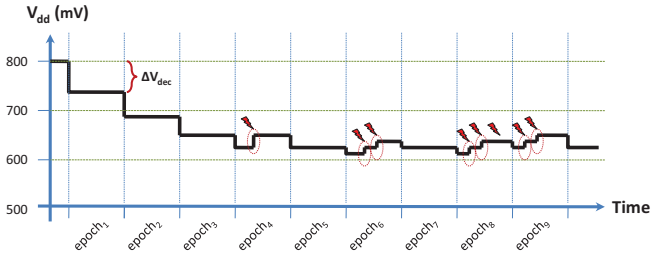


Figure 1: Changes to the V_{dd} of a router over time in Tangle.

the beginning of each epoch, V_{dd} is reduced by ΔV_{dec}. However, ΔV_{dec} is not constant; it is determined dynamically depending on the current V_{dd} value as follows:

$$\Delta V_{dec_i} = \begin{cases} 0mV & \text{if } V_i < 500mV \\ Max\{V_{MinStep}, \frac{V_i - V_{AvgTest}}{5}\} & \text{if } V_i > 500mV \end{cases}$$

where $V_{MinStep}$ is the minimum V_{dd} tuning step for the V_{dd} regulator. In this formula, if the V_{dd} for router i (i.e., V_i) is lower than $500mV$, we do not allow it to get any lower. The reason why we pick $500mV$ is that, as we will see, our process variation analysis shows that no router can operate correctly below this V_{dd}. For this low V_{dd}, we set the step size of V_{dd} increase (ΔV_{inc_i}) to $V_{MinStep}$.

For V_{dd} higher than $500mV$, ΔV_{dec_i} is determined dynamically based on the current V_{dd} of the router (i.e., V_i). In this formula, $V_{AvgTest}$ stands for the average sustainable V_{dd} of all the routers in the system, which is measured during the manufacturing-testing time. For instance, as we will see, given our variation profile, this $V_{AvgTest}$ is around $650mV$.

We can also derive $V_{AvgTest}$ without relying on measurements at manufacturing-testing time. In this case, we start the system at $800mV$ and use a fixed ΔV_{dec_i} equal to $V_{MinStep}$. Then, as the V_{dd} of the routers decrease, when they become stable after a certain number of epochs, we set $V_{AvgTest}$ to the average value of all of them. Once we have $V_{AvgTest}$, we can use a dynamic value of ΔV_{dec_i} according to the formula from that point forward.

Finally, for the higher values of V_i, we set ΔV_{inc_i} to be equal to ΔV_{dec_i} (we will discuss this aspect later). Hence, ΔV_{inc_i} also gradually shrinks, as we get closer to lower V_{dd} values, and finally becomes equal to $V_{MinStep}$.

The reason why we reduce ΔV_{dec} as we lower the V_{dd} of a router is to minimize the oscillations around the optimal V_{dd} value. We start to reduce V_{dd} fast and, as we get to lower values, we slow down the process. In Figure 1, in the first epoch, we reduce the V_{dd} of the router by around $60mV$, whereas in the next epoch we reduce it by $50mV$. However, ΔV_{dec} cannot be reduced beyond a certain threshold that is determined by the physical properties of the voltage regulator. During epoch 4, after we reduce the V_{dd} to around $620mV$, the system observes an error and we increase the V_{dd} back to its value before this epoch starts. In our design space exploration (Section 6.2.1), we show that setting the value of ΔV_{inc} to be equal to ΔV_{dec} gives us the best results. This is mainly because an equal value allows the complete cancellation of a wrong V_{dd} decrement. In epoch 5, no error happens. Nonetheless, in epoch 6, two errors happen, and we increase V_{dd} by two ΔV_{inc}. As will be described

in Section 3.5, because of our V_{dd} tuning algorithm, the V_{dd} of a router could be increased many times during an epoch. Thus, we introduce a threshold for the maximum number of times that the V_{dd} of a router can be increased in a single epoch. In Section 6.2.2, we sweep this parameter and decide to allow at most 2 V_{dd} increases per epoch. In epoch 8, we observe 3 errors. However, only 2 V_{dd} increases are performed during this epoch. Finally, in epoch 9, the V_{dd} of the router is increased again due to two errors.

As we can see, with Tangle, the V_{dd} of the routers converge to a state with tiny oscillations. Such oscillations are tolerable because the cost of error detection and tolerance is very small.

3.3. Error Detection

We merely rely on error detection because error correction is not an attractive choice for our application. This is because most error correction codes have the ability to fix a very limited number of errors in a word. For example, Single Error Correction Double Error Detection (SECDED), a form of Hamming codes which is widely used in memory systems, can only fix a single bit error in each word. In our system, as we reduce V_{dd}, there may be flits with multiple bit errors. Consequently, to guarantee correct operation, SECDED and other relatively simple ECCs would not be appropriate. We would need an ECC that has the ability to fix errors in several of the bits of a transmitting flit. Applying a strong ECC such as Orthogonal Latin Square Codes (OLSCs) [18] or BCH codes [5] is not practical for several reasons. First, the cost of the extra code bits that need to be transferred is comparable to the size of the flit itself. Second, the error checking and correction is too costly in terms of energy consumption. Third, the encoder and decoder for such a strong code have high complexity and area overhead [22]. Finally, the latency of such a complex error detection and correction would have a non-trivial impact on the performance of the system.

For these reasons, we employ error detection codes, which have lower overheads. Low-cost end-to-end error detection with Cyclic Redundancy Codes (CRCs) is attractive. CRCs are commonly used in embedded systems and networks for detecting data corruption [26]. However, we need to use a CRC polynomial that suits our design. We use WCDMA-8 with the following polynomial: $x^8+x^7+x^4+x^3+x+1$. This is divisible by $x+1$, meaning that it can catch any odd number of errors [26]. In addition, assuming a 100-core chip with a $1GHz$ clock and one message every 10 cycles per core, the system can operate without any undetected errors for thousands of centuries. This means that even a 7-bit CRC code might be enough for our purpose. However, we choose WCDMA-8 since there has been a lot of empirical and mathematical analysis on the strength of the code.

3.4. Voltage Tuning Invocation

When a destination node detects an error, it drops the message and waits for a retransmission from the source node. The source node has a watchdog timer for each message. A message is removed from the source node's outgoing buffer only when the response for it is received. Otherwise, after the watchdog timer reaches a certain value, the source node assumes that the message could not get to the destination because of an error, or that the message was faulty when it was received by the destination node. At this point, the source node sends a signal to the RMU to initiate a V_{dd} increase for the routers in the expected message path. While the V_{dd} of a router (e.g., R_j) is adjusted, the network is not disabled. Instead, the routers connected to R_j get a signal as if the incoming buffers in R_j are full and cannot accept any more flits. Hence, such routers temporarily buffer the flits to be sent to R_j.

3.5. Voltage Tuning Algorithm

When the RMU is notified of an error, it obtains the message path information from the source node. Given deterministic routing, the RMU knows what are the routers that the message was supposed to go through. Then, the RMU only increases the V_{dd} of the routers in that path. After the V_{dd} of these routers is increased by ΔV_{inc}, since there may still be faulty messages in the network, the V_{dd} of these routers is not increased again for a short period. The duration of such a *Hold* period is determined based on the average length of a path in the NoC, depth of the input/output buffers, number of virtual channels, and number of stages in a router (Section 5).

Figure 2 shows an example of how Tangle works for a 4x4 mesh NoC. In the figure, the routers with darker colors are slower due to process variations. Figure 2a shows the initial state, where all the routers receive the nominal V_{dd} of $800mV$. All messages are transmitted without errors. In the next few epochs, the V_{dd} of all the routers is reduced while there are no errors. Figure 2b shows the NoC when the V_{dd} of all the routers is $700mV$. At this point, when a message is transferred from $R_{1,1}$ to $R_{4,4}$, two faults happen along the way in the slowest routers $R_{1,4}$ and $R_{4,4}$. At the destination node, the CRC check detects an error and the RMU increases the V_{dd} of all the routers in this path by one ΔV_{inc} to $730mV$ (Figure 2c). Figure 2d shows the next epoch, where first the V_{dd} of all the routers is reduced by $20mV$. Then, a flit transmitted from $R_{3,1}$ to $R_{4,4}$ suffers an error in two routers. Therefore, the V_{dd} of all the routers in this path is increased by $20mV$. Next, Figure 2e shows a retransmission of the message. However, the V_{dd} of $R_{3,1}$ is still too low and an error occurs. Hence, the V_{dd} of all the routers in that path is raised by another $20mV$. Tangle continues to change the V_{dd} of the routers while gradually reducing the ΔV_{inc} and ΔV_{dec} so that the system becomes stable around the optimal V_{dd} for all the routers. Figure 2f shows the steady state, where all the routers operate at low V_{dd} while observing minimal error rates and performance loss.

Slow routers will be at the intersection of many faulty paths. This is shown in Figure 3. In the figure, the route that goes from $R_{1,1}$ to $R_{4,2}$ first fails. Hence the V_{dd} of all the routers in this path is increased. However, since the V_{dd} of $R_{2,2}$ is not high enough, the route that goes from $R_{2,1}$ to $R_{3,4}$ encounters an error. The overlap of these faulty paths causes the V_{dd} of $R_{2,2}$ to raise at a faster rate than in the other routers. Therefore, in Tangle, the V_{dd} of the slowest routers increases faster than that of other routers.

3.6. Multiple Routers per V_{dd} Domain

In a chip with current-technology voltage regulators, each NoC V_{dd} domain will include multiple routers — e.g., 8 routers. In this case, the Tangle algorithm remains largely the same. The

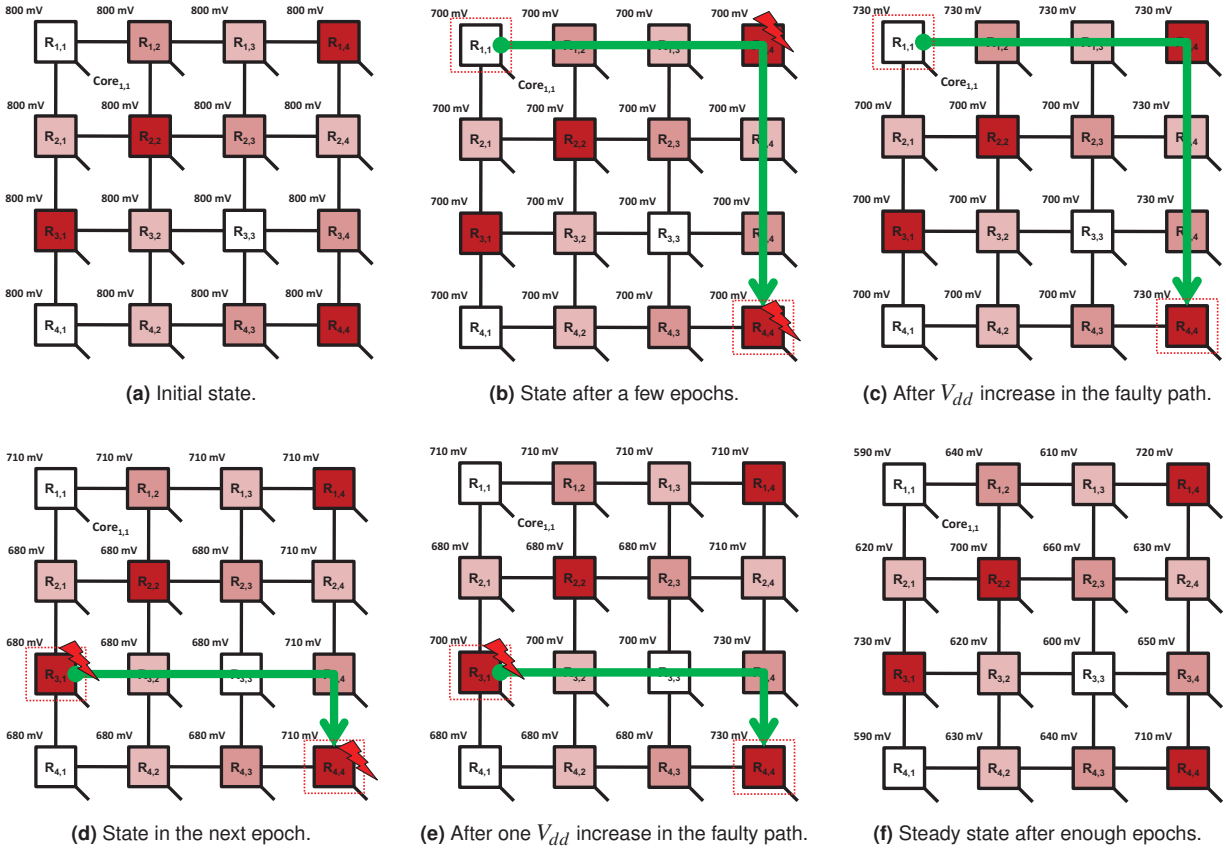


Figure 2: High-level example of Tangle's V_{dd} tuning algorithm for a 4x4 mesh NoC. Here, darker routers are slower due to process variations.

one change is that, when Tangle increases the V_{dd} of the routers in the path of a faulty message, it does so by increasing the V_{dd} of the *domains* traversed by the message. Such domains contain routers that were not in the path of the message.

With this design, the average steady-state V_{dd} of the routers ends up being higher than if each router had its own domain. This is because fast routers that could operate at a low V_{dd} now have to use a V_{dd} as high as the one needed by the slowest router in their domain. With higher steady-state V_{dd} values, the energy savings will be lower. For the same reason, the time to convergence to steady-state V_{dd} will be shorter. In all of our experiments, the V_{dd} always converged to a steady-state value.

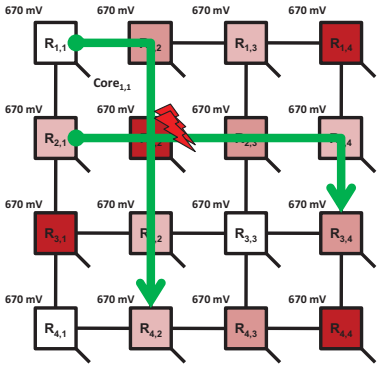


Figure 3: Slow routers are at the intersection of many faulty paths.

4. Tangle Design Aspects

4.1. Dealing with Contention

In workloads with high network traffic, there will be contention. Although long-duration message stalls are not very likely in NoCs [34], it is conceivable that the watchdog timer in a sender could time-out on a message because of contention. Consequently, in Tangle, when a timeout occurs, the system first checks that the message is not sitting in the network before it increases the V_{dd} of the domains in the path of the potentially-faulty message. Specifically, before the RMU initiates a V_{dd} increase, the RMU checks if the routers in the message path buffer the message locally (Figure 4). If they do, the RMU instructs the sender to reset the watchdog timer to give more time to the message delivery; otherwise, the RMU increases the V_{dd} in the domains and the sender resends the message.

4.2. Reliability Management Unit (RMU)

The RMU is a logic module connected to all of the V_{dd} domains in the NoC. At the beginning of each epoch, it sends a signal to lower the V_{dd} of all of the domains. In addition, when it receives information from a sender node on a timeout, it computes the message path and checks if the routers in the path have the message in their buffers. If they do, the RMU tells the sender to give more time to the message delivery. Otherwise, the RMU sends a signal to increase the V_{dd} of all of

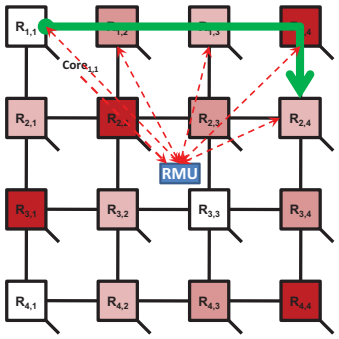


Figure 4: Handling timeouts due to congestion.

the domains in the message path, and tells the sender to resend the message.

The RMU uses a special network. The nework has very narrow links, since it only needs to transfer a few small control signals relatively infrequently. For scalability, in chips with many V_{dd} domains, the RMU network should be built in a hierarchical fashion with an H-tree structure (Figure 5). In the figure, each building block is a 4x4 mesh network with a local RMU. The local RMUs are connected to a global RMU.

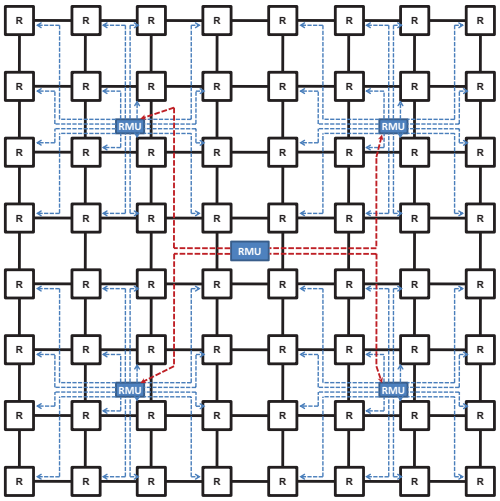


Figure 5: Hierarchical design of the RMU and its network.

4.3. Costs of Tangle Design

Tangle has four main costs that we analyze here.

4.3.1. Cost of Voltage Regulation. Tangle needs support for multiple V_{dd} domains. Voltage regulation induces a power loss of about 10% or more [24]. Consequently, to be conservative, we assume as baseline a conventional NoC without V_{dd} domains, and penalize our Tangle NoC with a 10% additional power. All the numbers reported in this paper for the power and energy consumption of a Tangle NoC always include this additional 10% power and energy overhead. In the future, we expect that, with a hierarchical, LDO-based V_{dd} regulator system, the losses will be smaller (Section 2.3).

4.3.2. Cost of Error Detection and Correction. Tangle adds an 8-bit CRC to each 128-bit flit (Section 3.3). There is energy and performance overhead associated with sending these codes over the network, encoding at the source node, and checking at the destination node. In our experiments, we account for these energy and performance overheads. Note, however, that the performance overhead of CRC usage is negligible: it only adds one extra cycle for the encoding, while the error checking is done in a shadow path speculatively.

4.3.3. Cost of the RMU and its Network. As per Section 4.2, the RMU operation is simple and the links of its network are very narrow. Hence, the RMU and its network use very little area. Moreover, since they are used relatively lightly, they add a negligible energy overhead. To deal with contention timeouts (Section 4.1), the RMU network can inspect whether a given message is present in a router. For this purpose, we have comparison logic in the router's input and output buffers. This logic is off the critical path. The comparisons to the elements in a given buffer are performed sequentially, without the need for additional buffer ports. Because of the relatively low frequency of these comparisons, of message retransmission, and of V_{dd} changes in general, we will see in the evaluation that the overall performance overhead of the RMU and its network is very small.

4.3.4. Cost of Message Retransmission. If we assume that a network without Tangle operates with absolutely no observed errors, then another cost of Tangle is retransmitting messages that suffered an error. Note that the number of errors induced by Tangle is small: the vast majority of the performance gains in Tangle come from exploiting the V_{dd} safety guardband, rather than from actually operating in the error region. Hence, while we model the energy and execution overhead of retransmissions, the impact is minor.

There are no explicit acknowledgment messages beyond the response messages provided by the cache coherence protocol. In typical protocols, there are always responses: a read request is followed by a response with the data requested; a write request is followed by a response with the write completion acknowledgment, and an invalidation is followed by an acknowledgment. In case the protocol does not provide a response for a transaction, we need to add it to Tangle. The requests are temporarily kept in buffers at the network entry ports. A buffer entry is deallocated when its corresponding response is received. If the response does not come within a certain time window, a timeout occurs. We model the energy overhead of these buffers as part of Tangle's overhead.

5. Experimental Methodology

To evaluate the potential of Tangle, we use a comprehensive methodology involving CAD tools for synthesis and critical path extraction, architecture-level systematic and random process-variation modeling, and microarchitectural simulation of a multiprocessor chip with an NoC for performance and energy. Details of each component follow, along with a description of the benchmarks.

We use the Verilog implementation of a virtual channel router from [41]. We modify this wormhole-switched router design to get a state-of-the-art 3-stage router similar to the one used in [40]. Next, the Synopsys Design Compiler is used to perform a timing analysis on this design. For each stage, we extract the 32 slowest paths and their netlists. These paths can potentially cause timing violations due to parameter variations.

We use a 3-stage router rather than a more aggressive single-cycle one for two reasons. First, single-cycle routers tend

Core Architecture Parameters	
Cores per chip; frequency	64; 1GHz
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)
BR target buffer; history table	1024 entries, 2-way; 2048 entries
Memory System Parameters	
L1 data cache	32KB, 2-way, 2 cycles latency, 64B line
L1 instr. cache	32KB, 2-way, 2 cycles latency, 64B line
L2 cache	32MB shared, static 64-bank addressing
	Bank: 8-way, 64B line, 6 cycles latency (local)
Main memory	260 cycles latency, 4 memory controllers
Tangle Network-on-Chip Parameters	
Topology; routing	8x8 2D-mesh; X-Y (and Y-X) wormhole routing
Number virtual channels	2 per each physical channel
Buffer depth in router	8 flits
Min. V_{dd} tuning step	$10mV$, 20 cycles latency
Max. V_{dd} tunings per epoch	2 times
Retransmission buffer depth	8 messages
Nominal V_{dd}	$825mV$ (includes guardband)
Length of an epoch	50,000 cycles
Num. routers per V_{dd} domain	16, 4, 1 (default)
Penalty to all Tangle NoCs	10% power and energy due to V_{dd} regulation
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})

Table 1: Default architecture and variation parameters. The memory hierarchy latencies refer to round-trip from the processor.

to operate at a slower clock rate, and single cycle per hop is actually achieved only when the load is low. Second, these routers often rely on speculative operation, which makes them less energy efficient [28].

To account for process variation, we use an updated version of VARIUS [42] called VARIUS-NTV [21]. Since we explore a future chip design, we perform the variation modeling at $11nm$. The baseline chip is composed of 64 nodes with private L1, an NoC with 64 routers, and 64 banks of a shared L2. To obtain the timing error rates due to process variation, we first generate the chip floorplan. Next, the logic efforts, which we extract from the synthesis phase, are used to enable a better delay modeling of the different stages of routers.

We use a cycle-level microarchitectural simulator of a multiprocessor chip with an NoC [40]. The simulator models both performance and energy consumption. As shown in Table 1, the baseline design of the network has 64 routers in an 8x8 2D-mesh. Each router has 5 physical channels (PCs), including the local port from the corresponding core to the router, and 2 virtual channels (VCs) multiplexed onto each PC for deadlock-free routing. A data message consists of six 128-bit flits and we use a buffer depth of 8 flits per VC. We model a wormhole-switched network with deterministic X-Y routing and credit-based flow control. For response messages, we use Y-X routing to overlap the forward and backward routes between a pair of source and destination nodes. The router, along with the proposed modifications, is implemented in structural RTL Verilog and synthesized using the Synopsys Design Compiler with Nangate $45nm$ open cell library. It is then scaled down to $11nm$ based on rules given in [2] and technology parameters from the ITRS report [19].

The simulator also models the processors and caches. The processors are 2-issue out-of-order Alpha DEC EV4-like cores. The frequency of the cores is set to $1GHz$, to save power and enable full-system operation under a typical power envelope.

We explore different network configurations. For example, we perform experiments with 16, 4, or 1 router per V_{dd} domain in the network. Given that we are examining a future chip design, the default system in our evaluation has 1 router per V_{dd} domain. We also vary the size of the network from 4x4 nodes to 10x10 nodes for scalability analysis. For each network size, we employ VARIUS-NTV to generate chips with representative parameter variation profiles.

After the V_{dd} of a router has been increased, its V_{dd} *Hold* period (Section 3.5) is computed as follows. For an 8x8 mesh network, it is known that the average length of a path is $\approx$16/3. Therefore, the expected period until faulty messages are drained from the network with no contention is: 2 (number of VCs) $\times$ 8 (maximum buffer depth per VC) $\times$ 3 (number of router stages) $\times$ 16/3 (average length of a path) = 256 cycles. We conservatively set the *Hold* period to 300 cycles.

The dynamic and leakage power numbers are generated using Orion [49] or, for some important measures such as the power in the buffers of the routers, using our synthesis experiments. Recently, there have been some concerns about the accuracy of Orion's results [16, 46]. However, these concerns primarily target Orion's power consumption in router buffers, and Orion's router area estimation. We do not use either of these estimations. For the power in the router buffers, our synthesis experiments estimate about $6.54mW$, compared to the $6.93mW$ reported in [46] as generated by SPICE.

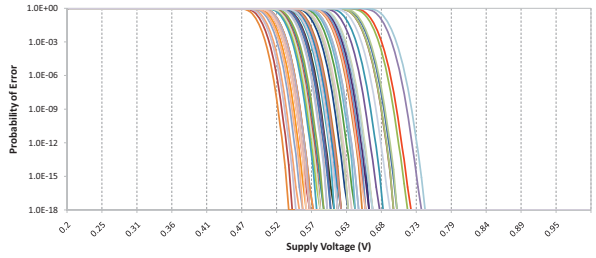
For our experiments, we run 15 multi-programmed workloads. They are a subset of those used in [34]. Of these workloads, 3 are commercial (sap, sjas, and tpcw), 9 are engineering (429.mcf, 433.milc, 436.cactusADM, 437.leslie3d, 450.soplex, 459.GemsFDTD, 470.lbm, 471.omnetpp, and 483.xalancbmk) and 3 are scientific (art, ocean, and swim). We use SimPoint [43] to select representative windows of instructions for simulation. For each workload, we use 8-15 simpoints. As a result, each core executes at least 5M instructions after warm-up.

6. Evaluation

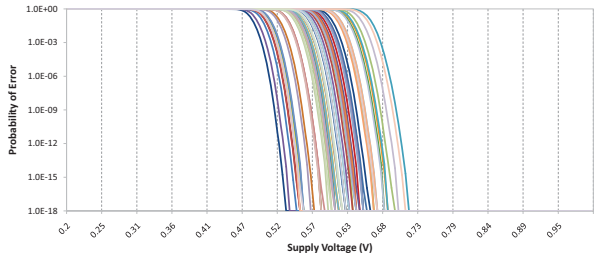
In this section, we start by evaluating different aspects of the Tangle design. Then, we perform a design space exploration and, finally, analyze the sensitivity of our results to several circuit and technology-based parameters.

6.1. Main Results

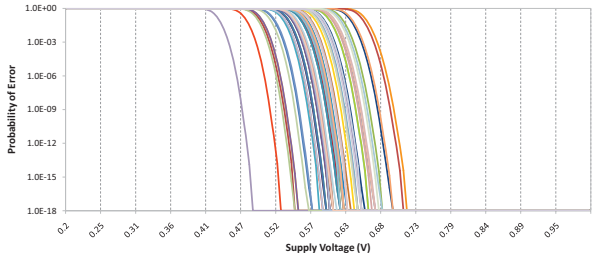
6.1.1. Probability of a Timing Error. Figure 6 considers a chip with a Tangle NoC and a given process variation profile. It shows the probability of a timing error in each of the three pipeline stages of the 64 routers, as a function of the V_{dd} applied. Figures 6a, 6b, and 6c correspond to the first, second, and third stages of the pipeline, respectively. In a figure, each curve corresponds to a particular router in the NoC.



(a) Probability of error as a function of V_{dd} in the first stage.



(b) Probability of error as a function of V_{dd} in the second stage.



(c) Probability of error as a function of V_{dd} in the third stage.

Figure 6: Impact of V_{dd} on the probability of a timing error in a particular stage of the router pipeline across all 64 routers of the NoC.

In the figure, the Y-axes are logarithmic. Hence, there is a rapid increase in the probability of a timing error as we reduce the V_{dd}. We can observe that process variations induce a wide distribution of the minimum error-free V_{dd} across the chip. For V_{dd} values above $750mV$, all the 64 routers are largely free of error; the chance of error in any stage of any router is less than 10^{-18} errors per cycle, which is equal to a few decades of operation without errors. However, some routers can operate in a V_{dd} as low as around $550mV$ without errors. Finally, as can be seen in the figure, the first stage of the router pipeline is the one that imposes the most restrictions on the timing requirements. The reason is that it has relatively longer critical paths.

6.1.2. Voltage of Routers over Time. Figure 7 shows the V_{dd} of 4 routers under Tangle over time. The plot also shows a curve for the average V_{dd} of all of the 64 routers in the NoC. This experiment was conducted while running the sap workload. The X-axis is shown in number of epochs. The initial V_{dd} of all the routers starts at $825mV$. The V_{dd} reduces rapidly during the first few epochs. Then, the V_{dd} step size gradually shrinks to limit the fluctuations around the optimal values. After around 30 epochs, the V_{dd} of each router becomes stable, with small oscillations. At that point, one of the routers ends up operating below $600mV$, while another operates above $700mV$. The average V_{dd} of the routers converges to $650mV$, which is a significant reduction from the initial V_{dd} value.

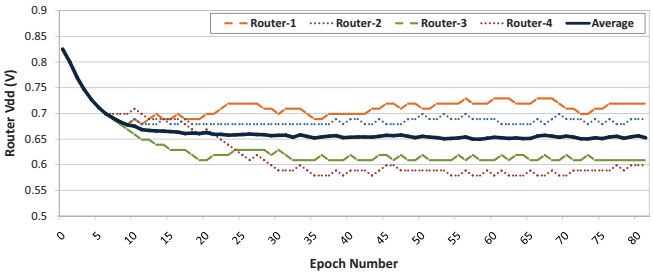


Figure 7: V_{dd} of different routers stabilizing over time. There is wide variation in the final V_{dd} of different routers across the system.

Figure 8 shows a snapshot of the V_{dd} map of the routers in our 8x8 Tangle system, running sap, after the V_{dd} values stabilize. The systematic component of process variation causes a spatial correlation in the V_{dd} of the routers. As can be seen, Tangle enables the routers to operate at a wide range of V_{dd} values, ranging from $590mV$ to $730mV$. This wide range provides an excellent opportunity for power savings. If we used a single V_{dd} for the whole NoC, the system would have to operate at $730mV$. It is also interesting to compare these V_{dd} values to the curves in Figure 6. We see that Tangle attains V_{dd} values that are close to the minimum V_{dd} values of the routers.

730	630	650	700	690	680	620	720
630	670	600	620	660	680	600	590
660	650	680	660	670	640	660	660
700	590	610	630	680	610	630	690
680	640	660	610	640	600	690	730
640	650	640	620	670	610	630	660
630	620	600	610	710	630	650	630
700	660	670	670	670	680	640	660

Figure 8: V_{dd} of the routers in an 8x8 Tangle NoC after the V_{dd} stabilize. The lighter colors show lower V_{dd} values.

In all the experiments, we observe that Tangle settles the V_{dd} in just a few milliseconds. Application phases tend to be at least $100ms$ long [43]. This gives Tangle ample time to stabilize the V_{dd}. Furthermore, temperature changes occur at the granularity of seconds. Hence, Tangle can easily adapt V_{dd} to temperature changes as well.

6.1.3. Energy Savings for Different Voltage Domain Sizes. The primary objective of Tangle is to reduce the energy consumption of the NoC by variation-aware reduction of V_{dd} without changing the frequency. Figure 9 shows the relative energy consumption of the Tangle NoC over a conventional NoC (i.e., one with no V_{dd} reduction and no V_{dd} domains) for different applications. In the figure, we keep the NoC fixed with 64 routers, and vary the size of the V_{dd} domains, from 1 router per domain to 4, and to 16. Recall that we penalize the Tangle NoC with a 10% extra energy due to V_{dd} regulation inefficiencies.

The figure shows that Tangle attains substantial energy reductions for all the workloads. For the future chip with one router per V_{dd} domain, Tangle reduces the energy by 28% on average. As we increase the V_{dd} domain size, the energy saved decreases. The reason is that, when an error is detected,

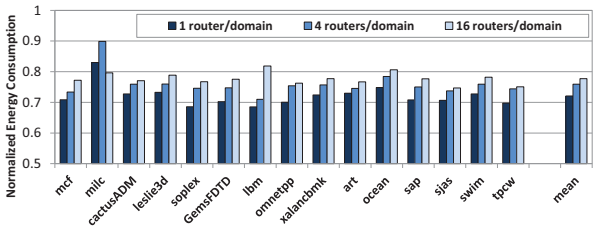


Figure 9: Normalized energy consumption of a 64-router Tangle NoC with different numbers of routers per V_{dd} domain.

Tangle increases the V_{dd} for a larger number of routers. This causes some routers to have a higher V_{dd} than they strictly need. Nonetheless, Tangle is robust in the presence of multiple routers per V_{dd} domain. For 16 routers per domain, which can be easily supported with current technology, the average energy reduction of Tangle is 22%.

6.1.4. Energy Savings for Different NoC Sizes. Figure 10 shows the relative energy consumption of the Tangle NoC over a conventional NoC for different mesh NoC sizes. The figure shows bars for NoCs with 16, 36, 64, and 100 routers. Each router is associated with one processor. We use our default of one router per V_{dd} domain.

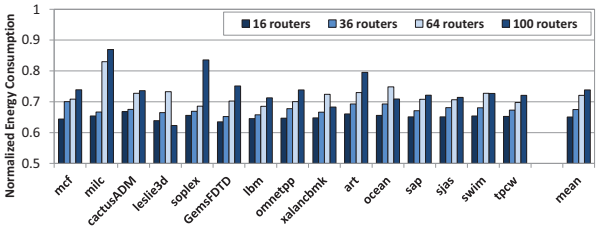


Figure 10: Normalized energy consumption of a Tangle NoC with different numbers of routers.

As shown in the figure, Tangle obtains substantial energy reductions across different NoC sizes. The average energy saved for 16, 36, 64, and 100 router NoCs is 35%, 32%, 28%, and 26%, respectively. We can see that Tangle is a scalable scheme. However, the relative energy savings reduce as the NoC increases in size. The reason is that the average distance between communicating nodes increases with the NoC size. This means that, on an error, Tangle will modify the V_{dd} of a larger number of routers. Effectively, the granularity of V_{dd} tuning becomes coarser, and there is less control over the exact V_{dd} of each router. To overcome this issue, in very large NoCs, we can split long paths into several shorter paths, and perform error checking more frequently as a message travels in the network. For instance, if a path from source to destination is 20 hops, we can split it into two paths and perform the error checking at the tenth hop and at the destination.

6.1.5. Performance Overhead. Tangle can add performance overhead to the execution — the result of message timeouts and retransmissions, and of periods of stall due to V_{dd} tuning. This overhead, however, is very small. Figure 11 shows the execution time of the applications using the Tangle NoC normalized to the execution time using a conventional NoC. The figure shows the data while varying the number of routers in the NoC from 16 to 100. We can see that the average performance overhead of Tangle for all the NoC sizes is around 1%. This is a negligible overhead.

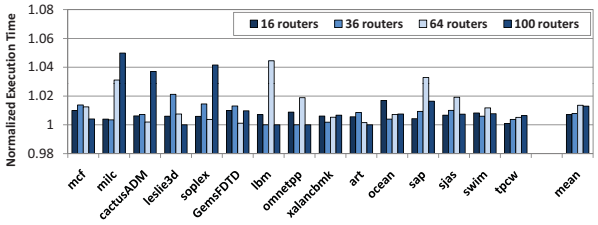


Figure 11: Normalized execution time of the applications running on a Tangle NoC with different numbers of routers.

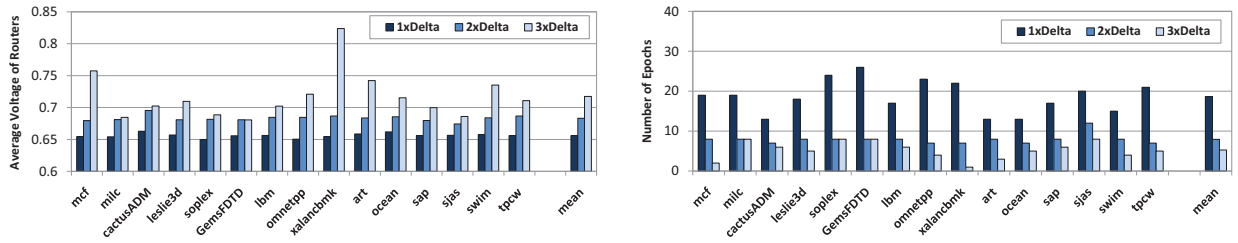
It can be shown that the performance overhead of Tangle NoCs with multiple routers per V_{dd} domain is also negligible.

6.2. Design Space Exploration

There are many hardware and policy parameters involved in the design of Tangle. However, due to space considerations, we only present the exploration of a few parameters that have especially interesting behaviors. They are: the size of V_{dd} step increases in a router, the maximum number of V_{dd} step increases allowed per epoch, and the use of variable- versus fixed-sized V_{dd} step changes. To reasonably determine the best value of each parameter, we study the impact of that parameter on the average V_{dd} of all the routers, after they stabilize in an 8x8 Tangle mesh. Further, we study the impact of each parameter on the number of epochs needed for the Tangle system to converge to stable V_{dd} values.

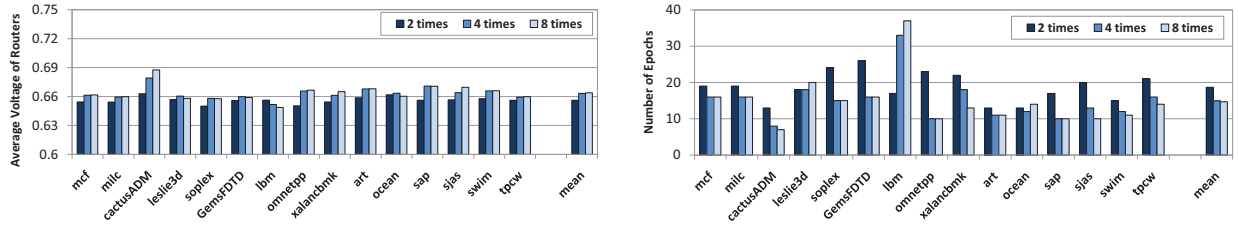
6.2.1. Size of Voltage Step Increases in a Router. In this section, we analyze the best ratio of ΔV_{inc} to ΔV_{dec}. This analysis is shown in Figure 12. Given a certain V_{dd} tuning epoch, we want to know, for a δ decrease of V_{dd} (at the beginning of the epoch), how many δs we should increase the V_{dd} by, as soon as an error is detected. We varied the ΔV_{inc} from δ to 3δs in this experiment. As we increase ΔV_{inc}, V_{dd} values can be changed at a faster rate, and fewer steps are required to get to higher V_{dd} values. However, there is less control over the V_{dd} of each router. This results in higher average V_{dd} values (Figure 12a), although it reduces the number of epochs that it takes for the convergence of the algorithm (Figure 12b). In practice, our main concern is to keep the V_{dd} low, while the performance overhead of our design is already low enough that reducing the number of epochs needed for the convergence of the algorithm will not make any difference in performance. Consequently, we set ΔV_{inc} to be equal to ΔV_{dec}.

6.2.2. Maximum Number of Voltage Step Increases per Epoch. The next parameter that we study is the maximum number of V_{dd} step increases allowed in an epoch. As shown in Figure 13, we vary this parameter from 2 to 8. We did not try allowing only a single V_{dd} increase per epoch because, in such a scenario, the increase can at most undo the V_{dd} reduction that occurred at the beginning of the epoch — never bring the V_{dd} higher than that. This eventually results in intolerable overheads. From the figure, we see that, when we allow many V_{dd} increases per epoch, the average V_{dd} of the routers increases slightly (Figure 13a). This is because, due to the end-to-end error detection used, we may induce several consecutive V_{dd}



(a) Average V_{dd} of all the routers after they stabilize. **(b)** Number of epochs needed to achieve stable V_{dd} values.

Figure 12: Effect of varying ΔV_{inc} on the average V_{dd} obtained, and the number of epochs needed to obtain stable V_{dd} for the routers in an 8x8 Tangle NoC.



(a) Average V_{dd} of all the routers after they stabilize. **(b)** Number of epochs needed to achieve stable V_{dd} values.

Figure 13: Effect of varying the maximum number of times that we allow a V_{dd} step increase in an epoch.

increases for routers that are already operating above their optimal V_{dd} value. As the same time, when we allow many V_{dd} increases per epoch, the system takes relatively fewer epochs to converge (Figure 13b). This is mainly due to the fact that the set of V_{dd} values that Tangle will converge to is higher. Overall, we set the maximum number of V_{dd} step increases to two. The reason behind this is that we get a lower average V_{dd}.

6.2.3. Variable- vs Fixed-Sized Voltage Step Changes. The last design parameter that we study is whether to use variable-sized or fixed-sized V_{dd} step changes. With variable-sized changes, we start with a relatively large step (i.e., $50mV$ for ΔV_{inc} and ΔV_{dec}) and gradually reduce it as V_{dd} gets to low values. For our experiment, we set the minimum V_{dd} step change to $10mV$ for both approaches. Hence, the fixed-sized approach always uses $10mV$ changes. Through experimentation, we find that both approaches achieve almost the same average V_{dd} values after stabilization. However, as can be seen in Figure 14, the variable-sized approach reaches these V_{dd} values considerably faster. Hence, we use the variable-sized approach in our design.

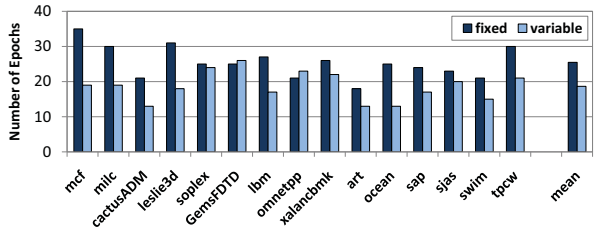


Figure 14: Number of epochs needed to achieve stable V_{dd} values with variable-sized or fixed-sized V_{dd} step changes.

6.3. Sensitivity Analysis

We now analyze the sensitivity of our energy and performance results to the V_{dd} tuning latency, the minimum V_{dd} tuning step size, and the V_{dd} guardband.

6.3.1. Voltage Tuning Latency. We study the sensitivity of Tangle to the latency of V_{dd} changes (in $10mV$ steps). We set the latency to 5, 20, 50 or 100 cycles. Based on what has been proposed in the literature, we use 20 cycles as the default latency for Tangle — although even faster approaches are being currently explored [24]. As we increase the latency to 50 and 100 cycles, we get only small changes in the energy savings and the performance overhead. Specifically, with 100 cycles, the energy savings of Tangle decrease to 26% of the conventional NoC energy (rather than 28%). In addition, the performance overhead over the conventional NoC reaches 2%.

6.3.2. Minimum Voltage Tuning Step Size. Figure 15 examines the impact of the minimum V_{dd} tuning step size on the energy savings of Tangle. We vary this parameter from $5mV$ to $20mV$. As the step size increases, the control over the V_{dd} of each router becomes coarser. This reduces the stability on the V_{dd} values across the system, and eventually results in higher V_{dd} values. Based on the state-of-the-art V_{dd} regulation techniques [27], we set the default minimum step size in Tangle to $10mV$. If we had to set the minimum step size to $20mV$, the energy savings of Tangle would decrease to 21% of the conventional NoC energy (rather than 28%). The performance overhead remains almost the same across the different V_{dd} tuning step sizes.

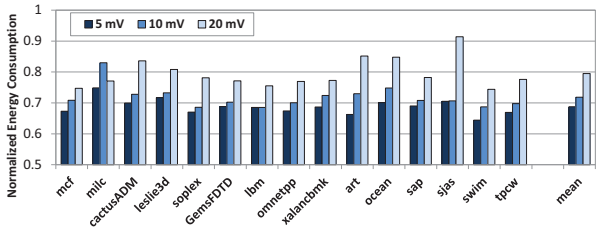


Figure 15: Varying the minimum V_{dd} tuning step in Tangle.

6.3.3. Voltage Guardband. Finally, we consider the impact of the V_{dd} guardband on the energy savings of Tangle. Fig-

ure 16 shows the energy consumption of the Tangle NoC over a conventional NoC for different V_{dd} guardbands. We consider guardbands of 5%, 10% (our default one), 15%, and 20%. As expected, the more conservative the conventional NoC is (i.e., the higher its V_{dd} guardband is), the higher the gains of Tangle are. For a 20% guardband, the average energy reduction of Tangle is nearly 40%.

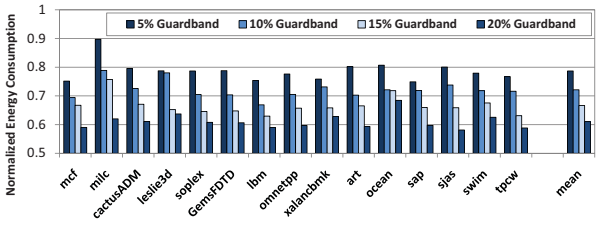


Figure 16: Normalized energy consumption of a Tangle NoC for different V_{dd} guardbands.

Figure 17 shows similar data when the 64-router NoC is organized with different numbers of V_{dd} domains. Specifically, it shows the energy consumption of the Tangle NoC over a conventional NoC for 1 router per V_{dd} domain, 4, and 16. To reduce the number of bars, the figure only shows the average across all the workloads. As expected, as we increase the number of routers per V_{dd} domain, the energy savings decrease slowly. Overall, however, Tangle's gains are robust across a variety of environments.

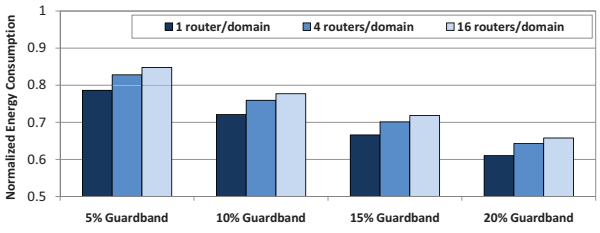


Figure 17: Normalized energy consumption of a 64-router Tangle NoC for different V_{dd} guardbands and different numbers of routers per V_{dd} domain.

7. Related Work

There have been many proposals for reliable NoC designs. Some have focused on the impact of wearout and process variations. Nicopoulos et al. evaluated the impact of process variations on NoCs with rigorous circuit analysis [37]. Their analysis does not include a fault model or the impact of faults at the system level. Vicis is a redundancy-based microarchitectural technique that provides system-level fault-tolerance for routers in an NoC when dealing with a few hard faults caused by gradual wearout [12]. Fu et al. consider both process variations and negative bias temperature instability (NBTI) effects on NoC router pipelines [13]. Li et al. studied the design of an NoC under process variations and realized that a high degree of process variations can force major design modifications to the underlying network architecture [29]. Ogras et al. explored the effectiveness of multiple voltage-frequency domains when dealing with deep sub-micron process variations in NoCs [39].

Orthogonal to these efforts, there are studies that focus on network links and try to compensate for the timing variations of the links by automatic detection and time borrowing or cycle tuning [33, 44]. Time borrowing or stealing is a technique that has been used to tackle process variations in the processor pipeline [30, 32, 47]. However, this class of techniques, if applied to routers, requires circuit-level modification to the underlying router design, and it is mostly done statically during manufacturing time. This prevents these techniques from adapting to the runtime conditions of the system. Additionally, for high degrees of process variations, these proposals require changes to the timing characteristics of the circuit.

A bus-encoding technique has been proposed to decrease the crosstalk between communication wires and prevent adversarial switching patterns [45]. In addition, many techniques have been proposed which look at the fault tolerance of both links and routers in the presence of permanent and transient faults [1, 9, 40]. This class of solutions incur higher overheads as they are proactive and deal with all types of faults in the same way. Instead, our goal is to adjust the circuit parameters to avoid these failures in the NoC. To enhance the energy efficiency and reduce the buffering requirements in an NoC, Kodi et al. advocate multi-purposing repeater logic on links as storage elements [25].

8. Conclusion

This paper presented a novel approach to save energy in an on-chip network in the presence of process variations. The proposal, called Tangle, monitors the errors of messages as they traverse the network and, based on the observations, dynamically decreases or increases the V_{dd} of groups of network routers. With Tangle, the different V_{dd} values applied to different groups of network routers progressively converge to their lowest, variation-aware, error-free values — always keeping the network frequency unchanged. This saves substantial network energy.

In a simulated 64-router network with 4 V_{dd} domains, Tangle reduced the network energy consumption by an average of 22% with negligible performance impact. In a future network design with one V_{dd} domain per router, Tangle lowered the network V_{dd} by an average of 21%, reducing the network energy consumption by an average of 28% with negligible performance impact.

References

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

[2] S. Borkar. Design Challenges of Technology Scaling. *IEEE Micro*, 19(4):23–29, July 1999.

[3] S. Y. Borkar. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. *IEEE Micro*, 25(6):10–16, 2005.

[4] S. Y. Borkar. Future of interconnect fabric: a contrarian view. In *SLIP*, 2010.

[5] R. C. Bose and D. K. Ray-Chaudhuri. On A Class of Error Correcting Binary Group Codes. *Information and Control*, 3(1):68–79, 1960.

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

[7] 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 *Symposium on VLSI Circuits*, June 2010.

[8] S. Dighe, S. R. Vangal, P. A. Aseron, S. Kumar, T. Jacob, K. A. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V. K. 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. *J. Solid-State Circuits*, 46(1):184–193, 2011.

[9] T. Dumitras, S. Kerner, and R. Marculescu. Towards on-chip fault-tolerant communication. ASP-DAC, pages 225–232, 2003.

[10] 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.

[11] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Architecture support for disciplined approximate programming. In *ASPLOS*, pages 301–312, 2012.

[12] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester. Vicis: A reliable network for unreliable silicon. DAC, pages 812–817, 2009.

[13] X. Fu, T. Li, and J. A. B. Fortes. Architecting reliable multi-core network-on-chip for small scale processing technology. In *DSN*, pages 111–120, 2010.

[14] 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 *Design Automation Conference*, June 2012.

[15] 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.

[16] M. Hayenga, D. Johnson, and M. Lipasti. Pitfalls of ORION-based Simulation. In *WDDD*, 2012.

[17] 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. *J. Solid-State Circuits*, 46(1):173–183, 2011.

[18] M. Y. Hsiao, D. C. Bossen, and R. T. Chien. Orthogonal latin square codes. *IBM J. Res. Dev.*, 14(4):390–394, July 1970.

[19] International Technology Roadmap for Semiconductors (ITRS),. *2012 Update*.

[20] F. Ishihara, F. Sheikh, and B. Nikolic. Level Conversion for Dual-Supply Systems. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, February 2004.

[21] 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. DSN, 2012.

[22] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe. Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding. MICRO, pages 197–209, 2007.

[23] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das. Design and analysis of an NoC architecture from performance, reliability and energy perspective. ANCS, pages 173–182, 2005.

[24] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *HPCA*, pages 123–134, 2008.

[25] A. K. Kodi, A. Sarathy, and A. Louri. iDEAL: Inter-router Dual-Function Energy and Area-Efficient Links for Network-on-Chip (NoC) Architectures. In *ISCA*, pages 241–250, 2008.

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

[27] S. Kose and E. G. Friedman. On-chip point-of-load voltage regulator for distributed power supplies. GLSVLSI, 2010.

[28] 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*, pages 63–70, 2007.

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

[30] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *MICRO*, pages 504–514, 2006.

[31] 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*, 28(1):60–68, Jan. 2008.

[32] X. Liang, G.-Y. Wei, and D. Brooks. Revival: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency. *IEEE Micro*, 29(1):127–138, Jan. 2009.

[33] 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*, pages 292–303, 2009.

[34] A. K. Mishra, N. Vijaykrishnan, and C. R. Das. A case for heterogeneous on-chip interconnects for CMPs. ISCA, 2011.

[35] T. N. Mudge. Power: A First-Class Architectural Design Constraint. *IEEE Computer*, 34(4):52–58, 2001.

[36] S. Murali, T. Theocharides, L. Benini, G. D. Micheli, N. Vijaykrishnan, and M. J. Irwin. Analysis of error recovery schemes for networks on chips. *Design and Test of Computers, IEEE*, 22:434–442, 2005.

[37] 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 Secur. Comput.*, 7(3):240–254, July 2010.

[38] M. Nourani and A. Radhakrishnan. Testing On-Die Process Variation in Nanometer VLSI. *IEEE Des. Test*, 23(6):438–451, Nov. 2006.

[39] Ü. Y. Ogras, R. Marculescu, and D. Marculescu. Variation-adaptive feedback control for networks-on-chip with multiple clock domains. In *DAC*, pages 614–619, 2008.

[40] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das. Exploring Fault-Tolerant Network-on-Chip Architectures. DSN, pages 93–104, 2006.

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

[42] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. VARIUS: A model of process variation and resulting timing errors for microarchitects. *Trans. on Sem. Man.*, (1):3–13, Feb. 2008.

[43] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. *ASPLOS*, 30(5):45–57, Oct. 2002.

[44] M. Simone, M. Lajolo, and D. Bertozzi. Variation tolerant NoC design by means of self-calibrating links. DATE, 2008.

[45] S. R. Sridhara and N. R. Shanbhag. Coding for system-on-chip networks: a unified framework. DAC, pages 103–106, 2004.

[46] C. Sun, C.-H. O. 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.

[47] A. Tiwari, S. R. Sarangi, and J. Torrellas. ReCycle: Pipeline adaptation to tolerate process variation. ISCA, 2007.

[48] A. Tiwari and J. Torrellas. Facelift: Hiding and Slowing Down Aging in Multicores. In *MICRO*, November 2008.

[49] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. MICRO 35, pages 294–305, 2002.

[50] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao. The impact of NBTI on the performance of combinational and sequential circuits. DAC, pages 364–369, 2007.

[51] 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*, pages 83–93, 2010.