Journal of
Low Power Electronics
Vol. 15, 115–128, 2019

Energy Efficient Network-on-Chip Architectures for Many-Core Near-Threshold Computing System

Chidhambaranathan Rajamanikkam*, J. S. Rajesh, Koushik Chakraborty, and Sanghamitra Roy

Dept. of Electrical and Computer Engineering, Utah State University, Logan, Utah, 84322, USA

(Received: 24 February 2019; Accepted: 26 April 2019)

Near threshold computing has unraveled a promising design space for energy efficient computing. However, it is still plagued by sub-optimal system performance. Application characteristics and hardware non-idealities of conventional architectures (those optimized for nominal voltage) prevent us from fully leveraging the potential of NTC systems. Increasing the computational core count still forms the bedrock of a multitude of contemporary works that address the problem of performance degradation in NTC systems. However, these works do not categorically address the shortcomings of the conventional on-chip interconnect fabric in a many core environment. In this work, we quantitatively demonstrate the performance bottleneck created by a conventional NTC architecture in many-core NTC systems. To reclaim the performance lost due to a sub-optimal NoC in many-core NTC systems, we propose BoostNoC—a power efficient, multi-layered network-on-chip architecture. BoostNoC improves the system performance by nearly 2× over a conventional NTC system, while largely sustaining its energy benefits. Further, capitalizing on the application characteristics, we propose two BoostNoC derivative designs: (i) PG BoostNoC; and (ii) Drowsy BoostNoC; to improve the energy efficiency by 1.4× and 1.37×, respectively over conventional NTC system.

Keywords: Near-Threshold Computing, Network-on-Chips, Energy Efficiency, Multi-layered NoC.

1. INTRODUCTION

Modern many-core chip design is plagued by barriers of prohibitive energy constraints and restrictive power budgets, while it is still expected to cater to the demands of diversified applications. *Near threshold computing* (NTC) comes as a saving grace to the energy-efficient computing paradigm by aggressively operating all computing platforms with a supply voltage close to the transistor threshold voltage. However, the tremendous increase in energy efficiency comes at the cost of a steep performance loss and performance variability (due to process variation).[1] Further, traditional many-core architectures designed to perform at nominal voltages yield sub-optimal performance at NTC. While a majority of existing literature focuses on optimizing the computing cores, research on the on-chip communication's impact at NTC has taken a back seat. In this context, we meticulously evaluate the application level and hardware performance characteristics of many-core NTC systems to specifically isolate the impact of the on-chip communication fabric—network-on-chip (NoC).

NTC circuits typically employ more devices to exploit application parallelism and compensate for the performance loss of a single device.[1] A direct consequence of this approach is the increased communication demand on the NoC owing to simultaneous interaction of many cores. This heightened communication demand, along with the following three prominent factors, delivers a severe blow to the on-chip communication latency and performance. First, we see an increase in the *inter-core packet hop distance* by virtue of an increase in the computational core count. Second, the supply voltage scaling to near threshold results in a *massive reduction of the NoC operational frequency*. Finally, the unavoidable effects of *process variation* (PV) presents a tremendous challenge in NTC systems. In this work, we demonstrate that the *traditional on-chip communication fabric creates a severe performance bottleneck in NTC systems*. In addition, we seek a solution to regain the lost performance without compromising on the energy efficiency of the system.

Contemporary research on NoC topology and architectures such as clustered NoC,[2] hierarchical NoC[3] and

*Author to whom correspondence should be addressed.
Email: cnaathan@gmail.com

tile based NoC,[4, 5] have aimed to reduce the inter-core packet hop distance.While these works are an important step forward, they do not adequately address the challenges of reduced operational frequency and PV induced performance variation posed by the NTC regime. Hence, to improve the NoC performance without compromising on the energy efficiency, we propose *BoostNoC*—a power efficient, multi-layered NoC architecture that efficiently caters to the demands of many-core NTC systems. *BoostNoC* is made up of two architecturally homogeneous layers contrasted in their design characteristics. While one layer is optimized for power, the other is optimized to boost the NoC performance under high communication loads. *To the best of our knowledge, this is the first work to exploit the unique opportunity presented by the variation in communication load across epochs to efficiently boost the NoC performance in NTC regime*.

In the preceding version of this work,[6] we critically analyzed the factors affecting the performance in many-core NTC systems. Our analysis clearly demonstrated that, the NoC communication bottleneck played a critical role in the system's sub-optimal performance at NTC. We proposed BoostNoC—a multilayered homogeneous NoC architecture, to boost the NoC performance under high communication load, and at the same time largely sustain the power and energy benefits of an NTC system. In this manuscript, we build on our previous work, and carefully discuss key application sensitive design decisions. We perform a thorough analysis of the two BoostNoC design augmentations—power gated and drowsy routers, using key metrics such as peak power, system performance and energy efficiency. Our key enhancements exclusive to this manuscript, are as follows:

- We examine the router occupation over the duration of applications, to fuel our *BoostNoC* optimization design augmentations (Section 5).
- We propose *PG BoostNoC*—a design augmentation to power gate the unused routers in the *BoPeL* (Boost Performance Layer), to improve the peak power consumption, and energy efficiency of the *BoostNoC* architecture (Section 5.2).
- We thoroughly evaluate the *PG BoostNoC* augmentation considering the in-die process variations prevalent in near threshold systems, using metrics such as average packet latency, system performance, peak power and energy efficiency (Section 7).
- We present the difference between the two *BoostNoC* design augmentations—*PG BoostNoC* and *Drowsy BoostNoC* and correlate the obtained data with application characteristics to make recommendations based on application characteristics (Section 7.6).

2. RELATED WORK

Over the last decade, near threshold computing has been studied extensively, and a major share of these works focus on developing circuits and optimizing the computation and memory for a many core NTC system [1, 4, 7–9]. Dreslinski et al. identified some of the prominent challenges hindering NTC from entering main stream system design and proposed some preliminary directions.[1] Two key challenges have prevented us from fully leveraging the potential of near threshold computing: (a) *parametric variation* and (b) *performance loss*.[1]

Parametric variation: To fully understand the impact of PV and capture the increased sensitivity to PV at NTC, researchers have developed microarchitectural PV models.[7] Further, several innovative solutions, such as the use of PV tolerant memory structures,[8] use of multiple voltage-frequency domains,[4] use of single voltage and multiple frequency domains,[9] and computational core pipeline weaving,[10] among others, have been proposed to tackle the challenges arising due to PV.

Performance loss: To reclaim the lost performance caused by the reduction in operating frequency, contemporary works have proposed circuit-architectural solutions, such as device optimization by improving channel doping profile,[1, 11] re-organization of private and shared cache structure,[12] super-pipeling[13] and clustered architecture,[1, 4] But the most intuitive approach has been to increase the number of computational cores to exploit application parallelism.[4, 14] A direct consequence of this approach has been the tremendous increase in the on-chip communication demand. *While a handful of previous works recognize this increase in communication demand,*[14] *no previous work tackles the performance bottleneck arising as its aftermath*.

Contemporary works on NoC energy efficiency reveal that although power-gating of idle NoC routers can significantly reduce the static power, it comes at the cost of significant performance and energy overheads.[15–17] In applications with a high communication load, mere power-gating of idle routers incurs substantial wake-up overheads. Chen et al. have proposed a power-aware NoC, NoRD (node and router decoupling), that enables power-gating bypass to prevent the node's ability from transferring packets, thereby prolonging the router's idle period duration.[15] Farrokhbakht et al. have proposed an efficient and scalable method for power gating NoC routers that reduces the wake ups by leveraging the characteristics of deterministic routing algorithms and mesh topology.[18] Similarly, Matsutani et al. have proposed a sleep control-based look ahead routing that identifies the packet arrival two hops ahead and thereby reducing the wake-up delay and frequent short-term sleep of channels.[17] Samih et al. have proposed Router Parking technique for CMPs that selectively power-gates the routers attached to the idle cores. They have also proposed two router parking algorithms to ensure the impact on packet latency is minimal.[19] Bogdan et al. have previously studied workload characterization, and modeling to leverage NoC traffic to improve power/performance of many core systems.[20,21] *However, no prior work categorically addresses the shortcoming of on-chip communication in many core NTC systems.*

Our work in this paper advances the research in this domain and focuses on optimizing the on-chip communication in many-core NTC systems. We make the following specific contributions:
• Clearly demonstrate the performance bottleneck created by sub-optimal NoC architectures in manycore NTC systems.
• Propose and evaluate a multilayered NoC architecture, *BoostNoC*, to boost the performance, while largely sustaining the power and energy efficiency benefits in a many-core NTC system.
• Develop optimal variants of *BoostNoC*, by meticulously analyzing the application characteristics and the NoC network traffic in many-core NTC system.

3. MOTIVATION

In this section, we quantitatively assess the performance bottlenecks in a NTC many-core system. The performance of a many-core NTC has two major contributing factors: application level and hardware performance characteristics. To understand application level characteristics, we study the performance scalability of various applications under an *idealized hardware* in Section 3.1. To carefully understand the impact from hardware performance characteristics, we decouple two of its major components: off-chip memory latency (Section 3.2) and on-chip interconnect latency (Section 3.2). Our *rigorous experimental data clearly demonstrates that on-chip interconnect latencies are the most dominant performance bottlenecks in an NTC many-core system.*

3.1. Application Performance Characteristics

Application speedups from parallel execution are bound by the prevailing fraction of serial code and do not improve linearly with an increase in the computational core count. Since the fraction of serial code varies across applications, it is critical to understand this application level bottleneck when we comparatively analyze (super-threshold computing) STC and NTC systems.

Figure 1 shows the effective application speedups obtained when a representative set of parallel workloads (SPLASH2 benchmarks) are executed on *ideal hardware* by scaling the processor count from 1 to 128 cores. The evaluation methodology used for this analysis is presented in detail in Section 6. We observe that only a couple of applications in this diverse set of benchmarks, can effectively scale beyond 60 cores. Benchmarks like *radiosity*, *cholesky* and *barnes*, have nearly ideal speedup indicating very little overheads due to the serial portions of the code. Other applications like *water.sp* and *raytrace* have large portions of serial code. Deploying these applications in NTC systems with hundreds of cores will result in decidedly sub-optimal performance.

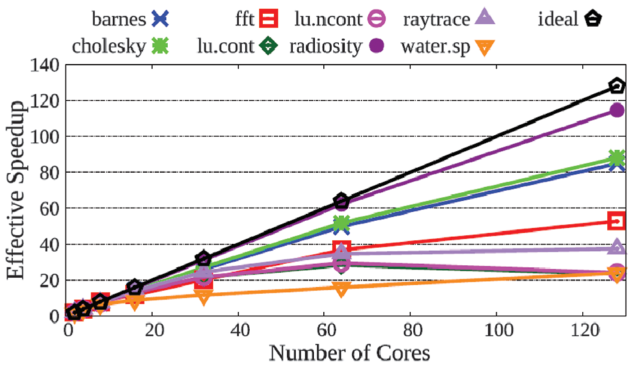


Fig. 1. Limitation due to application characteristics.

3.2. Hardware Performance Characteristics

To quantify the impact of notable hardware characteristics such as memory access latency and intercore communication on system performance, we consider a popular tile based 128-core architecture as our baseline NTC system.[4,22] The 128 cores are organized as 32 tiles (8×4) interconnected by a mesh network, with each tile consisting of 4 cores. We configure three systems and their parameters are shown in the Table I (Section 6 presents a detailed discussion of the methodology). The ideal system is configured with a unit latency for both off-chip memory access and on-chip communication. For interconnect bottleneck, we employed mesh network with a cycle latency for each hop between the nodes. Similarly, for memory bottleneck, we configured 10 cycle latency for each off-chip memory access. We use *Instructions Per Second (IPS)* as an accurate metric to evaluate the performance of these systems.

Memory Access: Figure 2(a) illustrates the performance degradation due to off-chip memory access latency in a 128-core NTC system. Our analysis proves that *memory access is not a prominent cause for performance bottleneck in NTC systems*. We observe that the average performance degradation due to memory access latency is a mere 0.7% and the highest degradation suffered is 1.5%

Table I. System configurations used to quantitatively analyze the performance bottleneck in many-core NTC systems.

Parameters	Ideal system	Interconnect bottleneck	Memory bottleneck
Architecture		Intel xeon processor ES series	
Cores		Tile-based 128 cores	
Voltage		0.35 V	
Frequency		200 MHz	
Technology		22 nm	
Memory latency	1 cycle	1 cycle	10 cycle
NoC latency	1 cycle	2D Mesh NoC (1 cycle/hop)	1 cycle

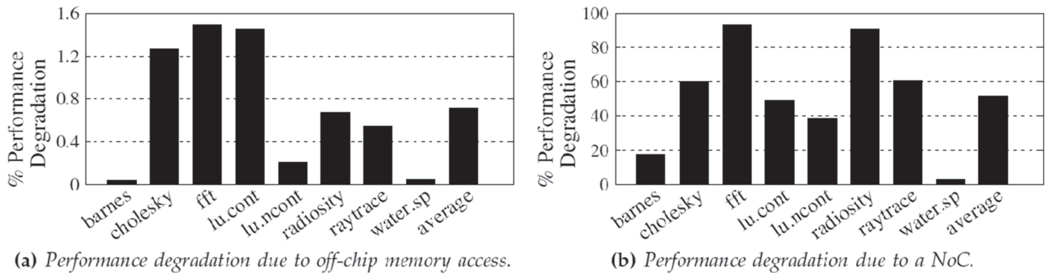


Fig. 2. Quantitative analysis of hardware characteristics to identify the cause of sub-optimal system level performance in many-core NTC systems.

for the *fft* application. The baseline is considered to be a 128-core NTC system with ideal memory access latency as shown in Table I.

On-Chip Communication: Figure 2(b) shows that the system performance degrades significantly due to the on-chip communication (network-on-chip) latency in a 128-core NTC system. Compared to an ideal system, the average performance degradation is a significant 50%, while *radiosity* and *fft* suffer from nearly 90% degradation in performance. Our evaluations reveal that the following three factors play a decisive role in the degradation in NoC performance.

• *Increase in communication demand*: When comparing the volume of packets injected in a 128-core NTC system to an isopower 16-core STC system, we found that the volume of injected packets increased by more than 3× in the NTC system. The rise in core count results in the increase of both inter-core, as well as, cores-memory communication.

• *Diverse latency distribution in NTC*: Figure 3(a) illustrates the distribution of communication latency in a 128-core NTC system. We observe that, on an average, more than 30% of the packets have a latency greater than 10 cycles. A similar analysis in the STC system showed that a mere 5% of the packets have a latency greater than 10. This diversity in latency distribution is the resultant of increased inter-core packet hop distance owing to a rise in the core count.

• *Reduced NoC operational frequency*: Figure 3(b) shows that the packet latency degrades by more than 6×, on average, in a tile-based 128 core NTC system. Applications such as *fft* and *radiosity*, suffer a latency degradation of nearly 16×. The increase in inter-core packet hop distance, along with the added detriment of reduced operating frequency, considerably increase the average packet latency.

3.3. Significance

The degradation in performance due to application (Section 3.1) and hardware characteristics (Section 3.2) help us characterize the demand in NTC systems. Our findings clearly demonstrate that *the on-chip communication is a severe bottleneck in many-core NTC systems*. Hence, we propose *BoostNoC*, a novel power-efficient NoC architecture for NTC systems to efficiently reclaim the lost performance.

4. BOOSTNoC ARCHITECTURE

4.1. Design Overview

We envisage a multi-layered NoC architecture, where the layers are architecturally homogeneous but optimized to contrasting design considerations. Our work in this paper demonstrates a novel incarnation of this concept—*BoostNoC*—that exploits the temporal nature of communication demand in NTC systems. The temporal nature refers to the variation of communication load across different epochs due to the inherent application characteristics.

Figure 4 illustrates the framework of our novel *BoostNoC* architecture. *BoostNoC* combines two architecturally homogeneous layers that are optimized to contrasting

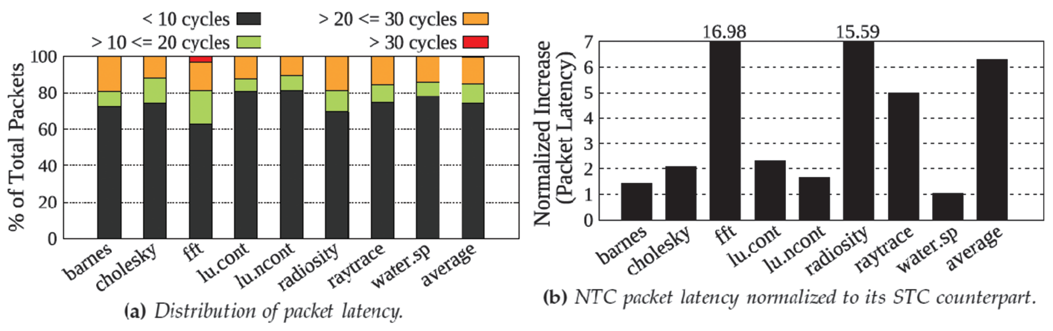


Fig. 3. Characterizing the loss in NoC performance in NTC. (a) Present the distribution of packet latency (in cycles) and (b) shows the degradation in packet latency in NTC systems.

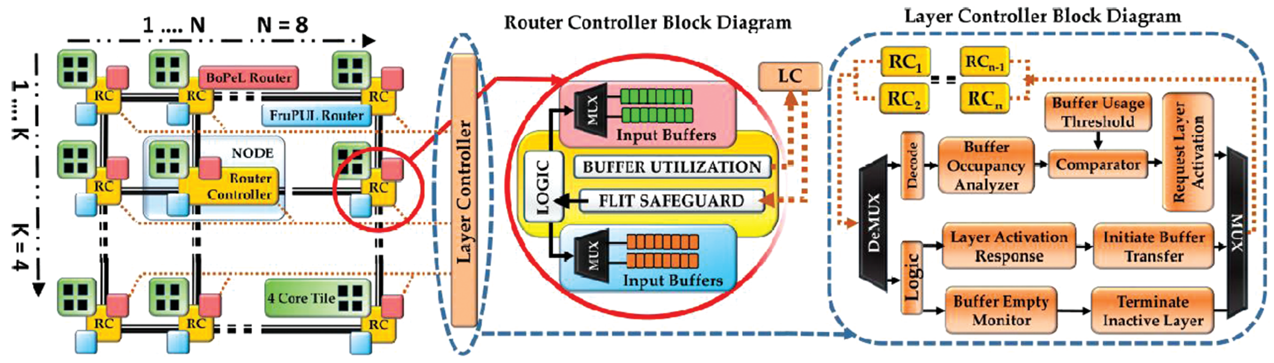


Fig. 4. BoostNOC architecture. The figure also shows the functional diagrams of the router and layer controllers.

design parameters. Based on the communication load, *BoostNoC* dynamically switches between the layers. While one layer is optimized for power efficient data transmission, the other layer is used to bolster the NoC performance. We detail the technicalities of *BoostNoC* in the following sections.

4.2. Temporal Communication Demand

Figure 5 shows the on-chip communication network utilization trend of 4 representative applications, running on a 128-core NTC system. The *x*-axis represents consecutive intervals during the application runtime. In most benchmarks, we see discernible patterns in the communication demand that fluctuates between epochs. In few epochs the cores are highly voluble creating a high load on the communication fabric, while in other epochs most cores are quiet (low communication demand). This temporal variation of network utilization can be correlated to the volume of injected packets experiencing long inter-core packet hop distance. Figure 6 illustrates this correlation for the *fft* benchmark. We see a sharp rise in network utilization in epochs with a high volume of long-distance packets.

Our novel *BoostNoC* architecture aims to exploit this temporal variation in communication demand by trading off chip area to bolster the NoC performance and energy efficiency.

4.3. BoostNoC Layers

Two architecturally homologous layers of NoC routers are interconnected in a mesh topology to frame the *BoostNoC* architecture. The two layers share the links

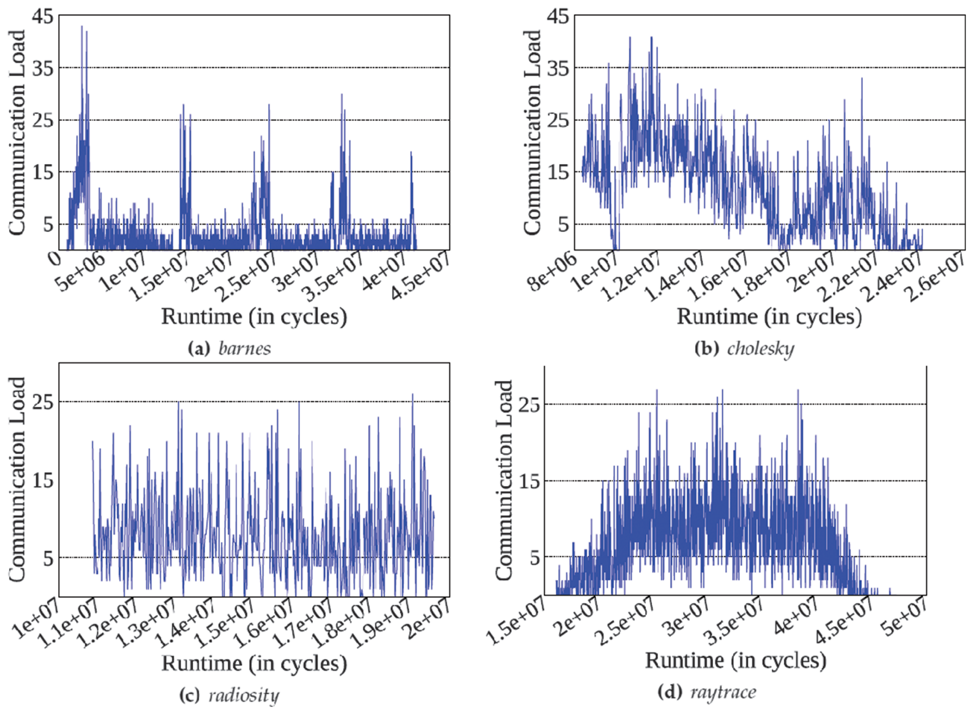


Fig. 5. Temporal variation of communication load for 4 different benchmarks. The plots illustrate network communication load (in %) during consecutive intervals of 2000 cycles for the whole application runtime. We see discernible patterns in all applications.

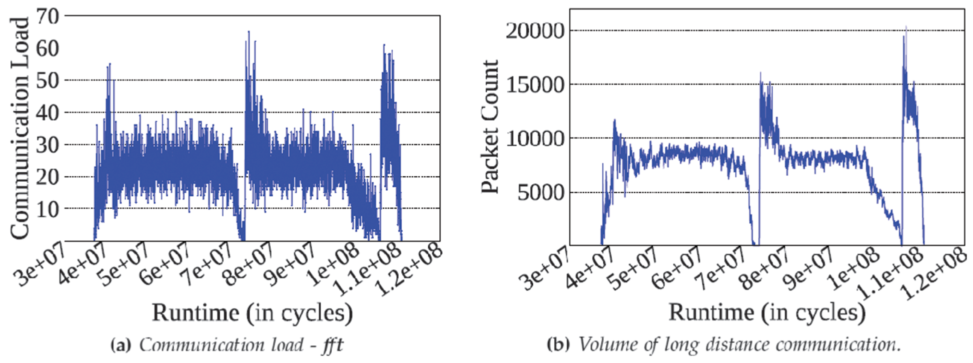


Fig. 6. Correlation between communication load and volume of long distance communication for the *fft* application. (b) shows the volume of long distance packets in consecutive epochs of 20000 cycles. The x-axis represents the application runtime in cycles.

between the routers as shown Figure 4. The two layers are:

Frugal power usage layer (FruPUL): The routers in this layer are optimized to operate in the near threshold voltage regime to provide power-efficient operation at a low communication load.

Boost performance layer (BoPeL): The routers in this layer are optimized to operate at the nominal voltage to bolster the NoC performance under a high communication load. The objective of *BoPeL* is to drain the in-flight packets at a quicker rate and offset the latency degradation caused by voluminous long distance communication.

At any given time, only one layer plays an active role in the communication fabric and the other layer is turned off. *FruPUL* is the default active layer as the cores are considered to be operating in the NTC regime. During epochs with high communication loads, *BoPeL* is activated (and *FruPUL* deactivated) to meet the demand and boost the NoC's performance. The layer switchover mechanism and the cost associated with it are discussed in Section 4.4.

4.4. Switchover Mechanism

Two switchover between the layers is the crux of the *BoostNoC* architecture. The primary constraint while switching between the two layers is to maintain lossless communication of packets while incurring minimal switching overheads. Figure 7 illustrates the process of switching between layers. Keeping the defined constraints in mind, we envisage four operational phases of the switchover mechanism explained below in conjunction with Figure 7.

• *Pre-initiate*: During normal NoC operation, one of the layers is active and the other is powered off. In this interval, the aggregate buffer occupancy of the routers in the active layer is carefully monitored. The buffer occupancy information serves as an indicator of the communication load on the network. It is the cardinal parameter behind the decision making process involved in switching between the layers. In Figure 7, we observe that *FruPUL* is active and the communication load is being monitored. When the load increases, the decision to switch to *BoPeL* is made.

• *Initiate*: Based on the decision, *BoPeL* is signaled to switch on. During the same time, all the routers in *FruPUL*

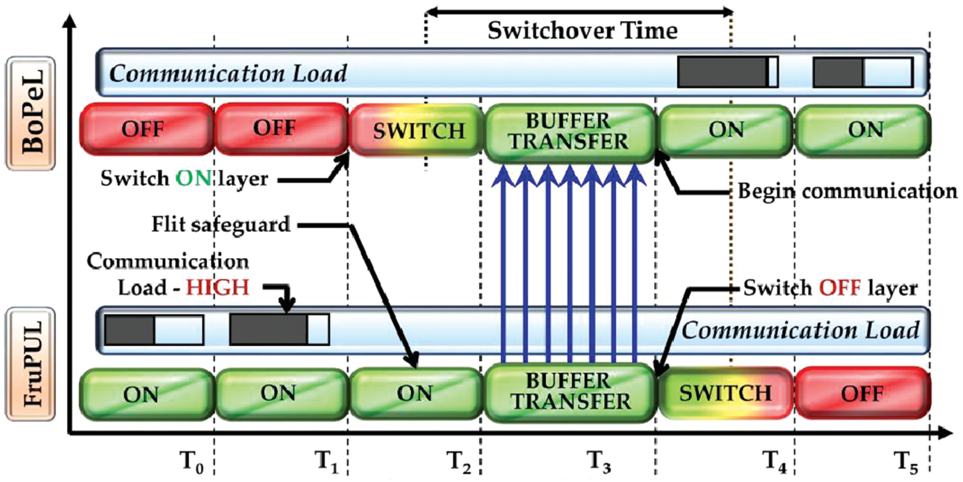


Fig. 7. Operational phases of the switchover mechanism.

are instructed to process the in-flight flits in each router and forward them to the input buffers of their respective downstream routers. The flits already present in each router's input buffers maintain status quo. We call this process *flit safeguarding*. The process of *flit safeguarding* is allowed to continue and complete until *BoPeL* (the other layer) is switched on and ready to handle traffic.

• *Transfer*: Once *BoPeL* signals ready, the packets in the input buffers of routers in *FruPUL* (one layer) are transferred to the corresponding routers in *BoPeL* (the other layer). The novel buffer content transfer mechanism overcomes the need to drain packets from the network and is elaborated in Section 4.5.

• *Terminate*: On receiving a signal from *FruPUL* that the buffer content transfer is successful and that all its buffers are empty, the layer is signaled to be powered off. Simultaneously, *BoPeL* is waved to begin normal operation.

4.5. Hardware Control Mechanism

Two *BoostNoC* architecture requires specific hardware enhancements to carry out its functions in an orderly fashion. We adopt two hardware control mechanisms known as *Layer Controller* and *Router Controller* to efficiently resolve and regulate the layer operations in the NoC. Each controller plays a definitive role to efficiently boost the NoC performance in NTC systems.

Algorithm 1 (Layer Controller Operation).

```
1: Initialize: Routers = $N$;          ▹Number of routers
2: Acknowledge: ack_LX
3: WaitForAcclimatizationPd();
4: for $k = 1 \to$ Routers do
5:    Evaluate BufferOcupancy(k);
6: end for
7: for $k = 1 \to$ Router do
8:    Evaluate RouterLocation(k);
9:    if ($BufferUsage >$ Usage$_{threshold}$) then
10:      RouterRequested++;
11:   end if
12: end for
13: if ($Router\ Requested > Router_{significant}$) then
14:      Enable reqactiv_LX;
15: end if
16: for $k = 1 \to$ Routers do
17:      Enable init_fs($k$);
18: end for
19: WaitFor respactiv_LX;
20: if (respactiv_LX) then
21:      Enable init_transbuf();
22: end if
23: WaitFor resp_bufempty;
24: if (resp_bufempty) then
25:      Enable term_LX(OLD);
26:      Enable begin_comm;
27: end if
```

Layer Controller (LC): The role of the layer controller is to monitor the network communication load by aggregating the information sent from individual router controllers. It functions like the brain of *BoostNoC*, and plays a central role in the decision process to switch between layers. Algorithm 1 shows the basic operation of the *LC*. As an initial setup, *LC* acknowledges the active layer (*ack LX*) and records the buffer occupancy of the routers in that layer (lines 1–6). It then continually monitors the average buffer occupancy information sent by individual router controllers during each epoch and based on the rules set in lines 7–15, it decides if a switchover in layer will yield a better outcome. In our experiments, we switch to BePeL layer when the average buffer utilization exceeds between 70% and 80%, depending on the buffer occupancy characteristics of the application. Once the decision is made to switch between layers, *LC* signals to turn on the alternate layer (*reqactiv LX*) and instructs the individual router controllers (*RC*) to trigger *flit safeguarding* (*init fs*). On receiving a response from the newly activated layer (*respactiv LX*), it instructs all *RCs* to begin inter-layer buffer content transfer (*init transbuf*) and waits for all *RCs* to signal for transfer completion (*resp bufempty*). At this point, the *LC* terminates the old layer (*term LX*), activates the new layer (*begin comm*) and goes back to monitoring the communication load.

Router Controller (RC): *RCs* are distributed agents with a three-fold functionality: (a) to sense local changes in the network, (b) to report gathered information to the *LC* and (c) to actuate responses when directed by the *LC*. Each individual *RC* reports its buffer occupancy to the *LC* at regular intervals (*report bufoc*) and waits for a decision. On receiving the *init fs* signal, the *RC* performs buffer content transfer as detailed in Section 4.5, reports successful transfer back to the *LC* and waits for *begin comm* to restart communication in the active layer.

Figure 8 illustrates the sequence of handshake signals between *LC* and *RC*, highlighting the operation of *BoostNoC*. *LC*, additionally ensures that once a layer is activated, it stays active for a set minimum period known as *acclimatization period*. The acclimatization period is added to amortize the cost associated with the layer switchover and to avoid the effect of thrashing between layers.

Inter-Layer Buffer Content Transfer: The router controller (shown in Fig. 4) plays a critical role in the inter-layer transfer of packets. The router in *FruPUL* is connected to its counterpart in the *BoPeL* using a bidirectional physical link controlled by the *RC*. The router in each layer consists of *n* buffers. Once the process of *flit safeguarding* is complete, *RC* evaluates the buffer occupancy of the active layer. The buffer contents of the active layer are serially copied to the buffers of the router in the alternate layer by selecting the appropriate MUX and DeMUX signals. A counter keeps track of all transactions between the two layers and once the value matches

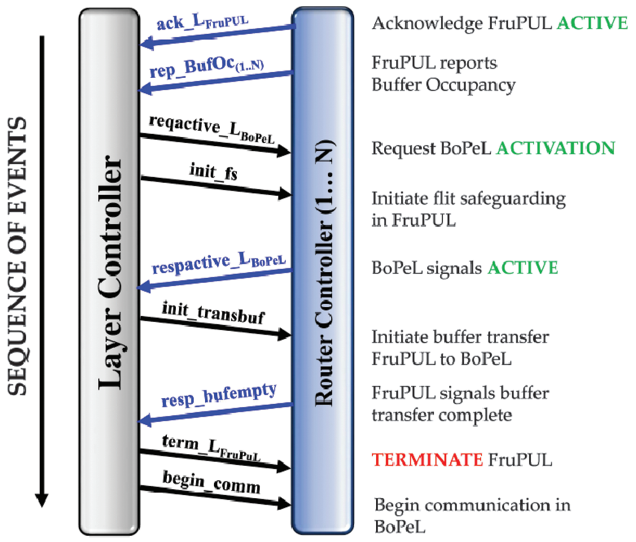


Fig. 8. Handshake communication between layer and router controller.

the buffer occupancy estimated before the process, the *RC* signals the successful completion of buffer transfer. This process happens simultaneously in the entire network. The serial transfer and transaction tracking between the two layers ensure a lossless transition between the two layers during a switchover. The cost associated with the switchover directly correlates to the buffer occupancy at the start of the process and the worst case switchover overhead depends on the buffer size of the routers.

5. ENERGY EFFICIENT BOOSTNoC ARCHITECTURES

In this section, we examine the traffic characteristics and network utilization of the *BoPeL* layer for various applications, to further optimize the *BoostNoC* architecture. Based on the key observations (Section 5.1), we propose two design derivatives of *BoostNoC—PG BoostNoC* (Section 5.2) and *Drowsy BoostNoC* (Section 5.3)—to further improve the energy-efficiency.

5.1. Key Observation

Breaking down the *BoostNoC* layer switching rule in Algorithm 1, we can state that *BoPeL* activation predominantly requires a high communication load on the network. Figure 9, presents an interesting observation of the *BoPeL* operation. In the majority of applications, a large percentage of the NoC routers remain idle intermittently, indicating that the communication load at any given time is spatially concentrated among a few routers. On an average, nearly 60% of the routers remain idle intermittently in each epoch of *BoPeL* operation, giving us the impetus to further optimize the *BoostNoC* energy efficiency. While the spatial concentration of communication load in *BoPeL*, provides the necessary information for *BoostNoC* optimization, it is not sufficient. On that account, we also examine the

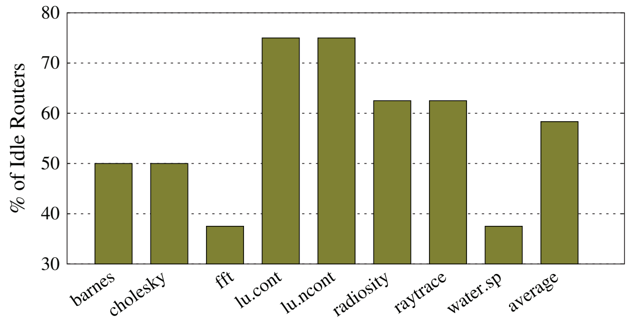


Fig. 9. Percentage of idle routers in BoPeL.

temporal distribution of communication in *BoPeL* to classify the *BoPeL* traffic trends into two categories, and propose *BoostNoC* derivative designs for optimization.

Sparse communication: The network traffic in *BoPeL* is temporally scattered, with only few routers sporadically exercised. For example, the *BoPeL* traffic characteristics of *lu.cont* in one epoch (Fig. 10(a)), reveal that the packets are quickly drained from the network, and for the rest of the epoch, the communication is sparse. During this period, the idle routers can be power gated with minimal effect on the NoC performance, giving rise to our first *BoostNoC* augmentation—*PG BoostNoC*.

Frequent communication: The network traffic in *BoPeL* gradually reduces, with a subset of the routers frequently being exercised. For example, *fft* endures a continued load, with frequent communication spikes as seen in Figure 10(b). Power gating routers is a sub-optimal design choice under these circumstances, due to the wait time associated with bringing the routers online. However, to extract optimal energy efficiency from the intermittent idle routers, we propose the use of drowsy SRAM as the input buffers—*Drowsy BoostNoC* (Section 5.3).

5.2. PG BoostNoC

The routers in the *BoPeL* operate at the nominal voltage, and hence have a significantly high power consumption. Idle routers in the *BoPeL* present an excellent opportunity to further improve the energy efficiency of *BoostNoC*. We employ power-gating of individual *BoPeL* routers on observing a drop in the communication load by the layer controller within the layer transition time to *FruPuL*. Router controllers that sense the local changes, decouples the idle routers from the network, signaling the surrounding routers about the change in the state. The state transition between OFF-ON/ON-OFF consume extra cycles and incur performance and energy penalties. We evaluate the performance and energy efficiency of power gating individual routers in *BoPeL* in Section 7.

5.3. Drowsy BoostNoC

The routers in the *BoPeL* operate at the nominal voltage and hence have a significantly high power consumption.

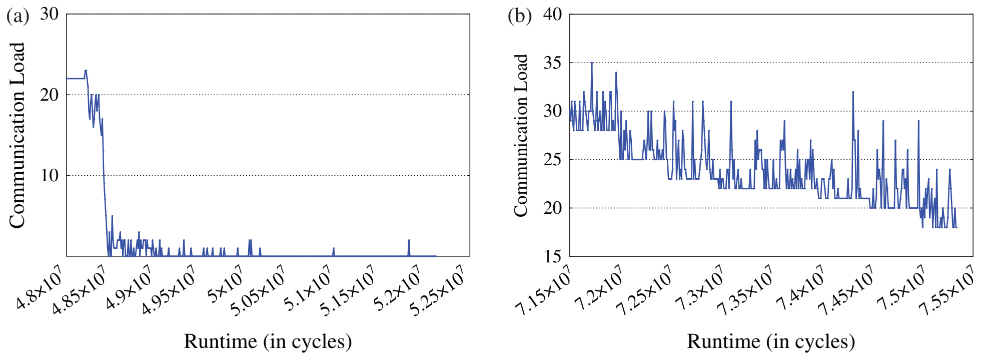


Fig. 10. Classification of network traffic characteristics for an epoch in BoPeL. (a) lu.cont benchmark. (b) *fft* benchmark.

By introducing *drowsy SRAMs* as buffers in the router, we add an additional low power operation mode to improve the energy efficiency. In this mode, a low voltage is supplied to the inactive routers, thereby reducing the leakage current. The idle routers are periodically put into *drowsy mode* and are woken up when the upstream router requests credit information. A single cycle cost is added to wake up a router in the *drowsy* state.[23] The decision to put the idle routers into the low power mode can be made by the router controller based on buffer utilization changes. We evaluate the improvement in energy efficiency due to *drowsy routers* in *BoPeL* in Section 7.

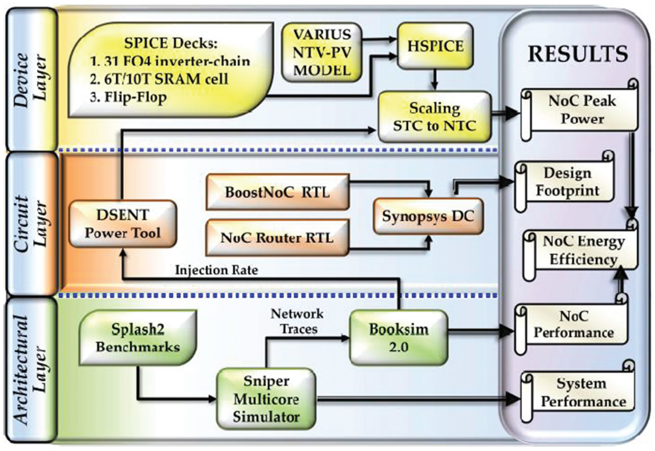
6. METHODOLOGY

Figure 11(a) presents the comprehensive cross-layer methodology we use to evaluate the efficacy of *Boost-NoC* architectures using three metrics: *peak power*, *performance* and *energy efficiency*. Architectural simulations are performed to assess the performance (Section 6.1), while the circuit layer analysis contributes valuable information regarding the design footprint and power characteristics (Section 6.2). Section 6.3 presents the procedure for device level analysis to obtain process variation parameters and STC to NTC scaling data.

6.1. Architectural Layer

Multi-core Simulation: We model an Intel Xeon E5 series processor on Sniper multi-core simulator[24] with the configuration shown in Table II. The STC system models 16 cores interconnected using a NoC (4×4 2D mesh topology). The NTC system models 128 cores in a tile based architecture interconnected using a 8×4 2D mesh NoC, with each tile housing 4 cores.[22] We use highly parallel large-set workloads from the Splash 2 benchmark suite to assess the performance of these systems and collect traces of the communication. We use booksim 2.0[25] to simulate and evaluate the NoC behavior. Splash 2 benchmark suite consists of parallel and well-diversified applications that can scale to 128 cores.[26] Chen et al. showed that the maximum delay deviation due to within-die PV is a colossal 200% for the NTC regime at 22 nm and thus cannot be discounted.[27] We therefore used this delay variation to model PV-affected NTC core.

NoC Simulation: We model a 8×4 2D mesh NoC mimicking a 32 tile-based NTC system on the Booksim



(a)

Parameters	STC Configuration	NTC Configuration
Architecture	Intel Xeon Processor E5 Series	
Cores	16	128
Voltage	1.0*V*	0.35*V*
Frequency	2.3GHz	200MHz
Technology	22*nm*	22*nm*

(b)

Fig. 11. (a) BosstNoC cross-layer methodology. (b) STC and NTC system configuration parameters.

Table II. Area overhead of BoostNoC and its derivatives (baseline is Always NTC).

Metric	BoostNoC	Drowsy-BoostNoC	PG-BoostNoC
Area overhead	12.4%	13.2%	16.0%

Simulator.[25] The router has a 4-stage pipeline of route computation, virtual channel allocation, switch allocation and switch traversal. We simulate the traces collected from Splash2 benchmarks and observe the NoC behavior and study various traffic characteristics. We implement the *BoostNoC* architecture with functionality detailed in Section 4 and evaluate the performance of the NoC. Our evaluation carefully considers the impact of PV on the NoC performance.

6.2. Circuit Layer

To estimate the design footprint and hardware overheads of our architecture, we augment the open source NoC router RTL[28] with the hardware control mechanisms discussed in Section 4.5. We synthesize the NoC router RTL using the 32 nm standard cell library using Synopsys Design Compiler. We use the DSENT power modeling tool[29] to determine the NoC leakage and dynamic power estimates considering the PV parameters evaluated in the device layer. The network and router configuration are identical in Sniper, Booksim, as well as, DSENT to maintain uniformity.

6.3. Device Layer

We obtain the 22 nm PTM model for HSPICE simulations and customize it in order to generate leakage and dynamic power behavior at STC and NTC regimes.[30] NTC circuits are highly susceptible to process variation. Our HSPICE evaluations model the effect of PV based on VARIUS-NTV[31] and we use these results while scaling from STC to NTC. The details of our scaling methodology follows:

Power Scaling from STC to NTC: Scaling the entire power from the STC to the NTC region presents a methodological challenge. HSPICE simulation of an entire NoC architecture is computationally intense. To manage the complexity, we scale the STC power to NTC using the following three categories.[32]

Combinational logic: This is scaled using the STC/NTC characteristics of the canonical 31 fanout-of-4 inverter-chain as the representing circuit.[14]

Storage elements: We scale the on-chip SRAM power by investigating the power scaling trend from the STC 6T SRAM cell to the NTC-friendly 10T SRAM cell.[33]

Interconnect: We estimate the interconnect power to be 50% of the dynamic power based on previous work.[32] Since scaling the supply voltage equally affects both interconnect power and dynamic power, we assume that their relative weight remains unchanged for STC and NTC.

7. EXPERIMENTAL RESULTS

In this section, we discuss the experimental results obtained from our simulation of the *BoostNoC* architecture and its energy-efficient variants considering the with-in die PV. Section 7.1 summarizes the different schemes that we use in our simulations for baseline comparison. We evaluate the effectiveness of our proposed architectures using three metrics *performance* (Section 7.2 and Section 7.3), *peak power* (Section 7.4) and *energy efficiency* (Section 7.5). In Section 7.6, we present a detailed trade-off analysis between the two variants of *BoostNoC*, based on the application characteristics. We conclude our results section by presenting the design footprint in terms of area overhead in Section 7.7.

7.1. Evaluation Schemes

We use five schemes to evaluate our proposed architectures. They are:

Always NTC: The NoC and the cores are both operated in the NTC regime throughout the application runtime. In theory, this scheme is extremely energy efficient however at the cost of a substantial degradation in performance. Moreover, the with-in die process variation significantly affects the performance/power characteristics of both the cores, as well as, the NoC in this scheme.

Always STC: In this scheme, the cores are operating in the NTC regime, while the NoC is operating at nominal voltage (STC). This configuration offers the best performance while taking a significant hit in peak power and energy efficiency. The cores substantially suffer from the effect of process variation. However, the NoC exhibits lower variation in performance/power characteristics as it operates at the STC regime.

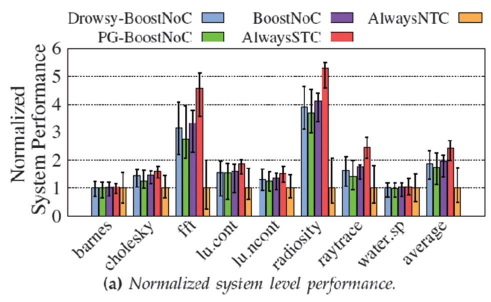
BoostNoC: Our proposed *BoostNoC* architecture, discussed in Section 4, uses two layers (*FruPUL* and *BoPeL*) to provide the best of both worlds. The architecture sacrifices chip area to deliver better performance and energy efficiency. The process variation affects both cores and NoC significantly. Since NoC operates in *FruPUL* layer during most of the application runtime, the effect of process variation is high compared to an *always STC* scheme.

PG BoostNoC: In this scheme, the unused routers in the *BoPeL* are power gated to reduce the static power consumption. As discussed in Section 5.2, the objective of this technique is to improve the NoC's peak power and energy efficiency, while taking a small hit on performance.

Drowsy BoostNoC: In this scheme, we employ drowsy SRAMs as the NoC router buffers (Section 5.3) in the *BoPeL*. The unused routers are able to improve NoC energy efficiency by transitioning into a drowsy state, with a minuscule wake up delay.

7.2. System Level Performance Analysis

Figure 12(a) shows the normalized system level performance of the *BoostNoC* architecture and its two variants,



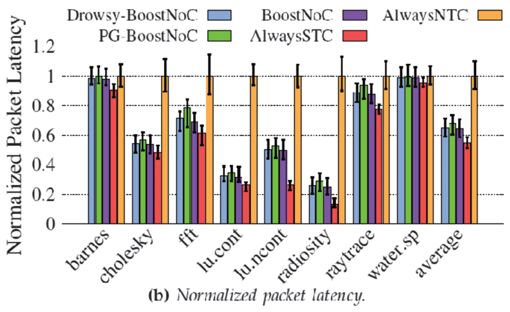


Fig. 12. (a) System level performance improvement of our proposed BosstNoC architectures normalized to Always NTC scheme (high is better). (b) Normalized reduction in packet latency of BoostNoC compared to Always NTC scheme (lower is better).

considering within-die process variation. The performance is normalized to the baseline PVfree *always NTC* scheme. Our results demonstrate that on an average, the *BoostNoC* improves the system performance by nearly 2×. Benchmarks with a high communication demand such as *fft* and *radiosity* show even higher performance improvement (nearly 4×). However, applications with low communication demand (*barnes* and *water.sp*) are less sensitive to the boost in operating frequency and hence deliver a small improvement in the system level performance.

Our evaluations show that the two variants of *BoostNoC—PG BoostNoC* (1.74×) and *Drowsy BoostNoC* (1.8×), slightly compromise on the system performance. This variation in performance is due to the additional time required for the routers to transition from sleep/drowsy state to the operational state. As expected, Figure 12(a) demonstrates that, *Drowsy BoostNoC* offers better performance than *PG BoostNoC*, due to the faster state transition times. We observe that the degree of variation in performance between the two variants are application dependent and is discussed in detail in Section 7.6.

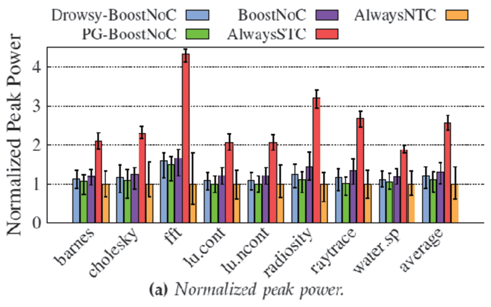
7.3. NoC Performance Analysis

Figure 12(b) illustrates the packet latency reduction due to *BoostNoC* and its variants. This reduction in packet latency directly translates into a system level performance improvement. Figure 12(b) demonstrates the on-chip communication performance as compared to the baseline PV-free *always NTC* NoC. On an average, *BoostNoC* improves the packet latency by nearly 40% compared to a conventional *always NTC* scheme. The performance of *Drowsy BoostNoC* is fairly identical to the *BoostNoC* architecture due to negligible overhead in the transition from *drowsy mode* to ON state. On the other hand, *PG BoostNoC*, suffers from larger state transition time (power gated OFF to ON), and hence achieves a slightly higher packet latency compared to *BoostNoC*. *Always STC* performs better than *BoostNoC* as the NoC operates at a higher frequency throughout the application runtime. Our results demonstrate that applications with high communication loads significantly benefit from the *BoostNoC* architecture, and, the application characteristics determine the performance variation between the two *BoostNoC* variants (Section 7.6).

7.4. NoC Peak Power Analysis

Figure 13(a) compares the peak power dissipated among the different simulation schemes. The values obtained are normalized to baseline PV-free *always NTC* peak power which is expected dissipate the least power. *BoostNoC* suffers from a nearly 30% rise in the peak power on an average, due to the switchover to *BoPeL* which operates at the nominal voltage. However, this is noticeably better than the *always STC* scheme, which incurs more than 2× increase in peak power.

Drowsy BoostNoC and *PG BoostNoC* experience peak power dissipation values of 20% and 10% over *always*



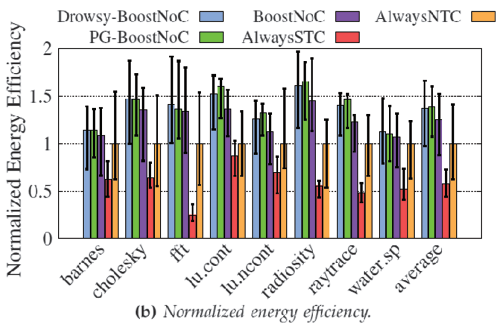


Fig. 13. (a) Normalized peak power of BoostNoC architecture compared to PV-free Always NTC (lower is better). (b) Energy efficiency of BoostNoC architectures normalized to PV-free Always NTC (Higher is better).

NTC, respectively, which is better than the peak power seen for *BoostNoC*. The improvement in peak power seen in *PG BoostNoC*, over *BoostNoC* is due to the reduction in static power by power-gating the idle routers in the *BoPeL*.

7.5. NoC Energy Efficiency Analysis

Figure 13(b) compares the normalized energy efficiency of the schemes. In a sense, *performance delivered per watt* is an accurate measure for comparison of the schemes as it accounts for both performance, as well as, power. Our analysis shows that, though the performance of the *always STC* scheme is significantly higher than other schemes, it is highly energy inefficient. The proposed *BoostNoC* provides a favorable trade-off between power and performance, and hence surpasses conventional NTC architectures by 25%. Both *Drowsy BoostNoC* and *PG BoostNoC* further improves the energy efficiency over *always NTC* by nearly 40%.

Water.sp has a low runtime and a low communication demand limiting the duration of operation in *BoPeL*. These characteristics of *water.sp* prohibits *BoostNoC* from improving its energy efficiency. Similarly, the meager improvement in *barnes* is due to its high compute and low communication attributes. Applications such as *Radiosity*, and *lu.cont* experience larger improvements in energy efficiency, leveraging the *BoostNoC* architecture well.

Figure 13(b) reveals an interesting consequence of the variations in application characteristics on the two design derivatives of BoostNoC. Section 7.6 presents a detailed analysis on why *PG BoostNoC* performs better for some applications, while *Drowsy BoostNoC* is favorable for others.

7.6. Analysis: PG BoostNoC versus Drowsy BoostNoC

While Figures 12 and 13 demonstrate the relative impact of *BoostNoC* compared to *always NTC* and *always STC*, Figure 14 primarily analyzes the variation in performance and energy efficiency between the design derivatives of *BoostNoC*. *PG BoostNoC* and *Drowsy BoostNoC* differ primarily in the operation in BoPeL. We observe that application characteristics play a key role in deciding an optimal choice of *BoostNoC* architecture. We correlate the results in Figure 14 to the following three key application characteristics:

Communication load: The duration *BoostNoC* operation in BoPeL is a direct resultant of the high communication load in applications. Applications such as, *fft*, *radiosity* and *raytrace* process a large percentage of the total packets in BoPeL, thereby seeing a larger impact due to *BoostNoC*. Figure 12(a) illustrates the relative impact of *BoostNoC*, and clearly reveals that *BoostNoC* seizes the opportunity to reclaim the lost performance of *always STC*. Applications such as *barnes* and *water.sp* have relatively low communication load, echoing the data that for higher communication load, the *BoostNoC* architecture is more effective.

Communication Frequency: The temporal distribution of packets impact the transition between *on* and *drowsy/off* states, thereby affecting both energy and performance. Frequent arrival of packets favor the use of drowsy SRAMs, over power gating, due to the wake up delay overheads associated with power gating the routers. Communication patterns of *cholesky*, when compared with *radiosity* (Fig. 5) serve as a good example to analyze the two *BoostNoC* designs. The spike in network activity in *radiosity* is even, and temporally well distributed, whereas in *cholesky*, we see varying network activity with frequent network spikes. In Figure 14(a), the larger difference in performance between *PG BoostNoC* and *Drowsy BoostNoC*, in *cholesky* as compared to *radiosity* emphasizes the impact of communication frequency.

Idle Routers: The spatial distribution of packets in the BoPeL also play a significant role in the NoC efficiency, as it determines the number of routers that can be put to drowsy/off state. Correlating Figure 9, with Figure 14(b), we observe that for applications that have a higher percentage of idle routers in the BoPeL, *PG BoostNoC* has an edge over *drowsy BoostNoC* in efficiency. For *fft* and *water.sp*, which have the lowest percentage of idle routers in BoPeL among our tested applications, *Drowsy BoostNoC* is clearly the better choice between the two *BoostNoC* derivative designs.

7.7. Design Overheads

The overheads due to the cost associated with switching between layers is accounted for in our performance

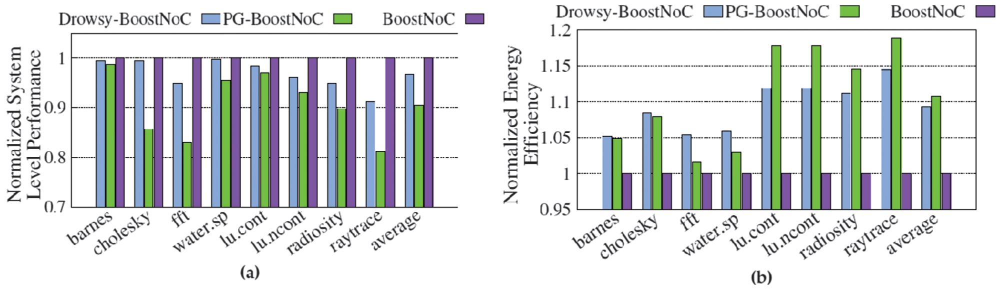


Fig. 14. Analysis between drowsy-BoostNoC, PG-BoostNoC and BoostNoC. (a) Normalized system level performance (higher is better). (b) Normalized energy efficiency (higher is better).

evaluations. *BoostNoC sacrifices chip area to deliver better performance and energy efficiency*. Table II demonstrates the area overhead of *BoostNoC* and its derivatives. The relative chip area increases by 12% as *BoostNoC* consists of two architecturally homogeneous layers of NoC. Similarly, the relative chip area for *Drowsy-BoostNoc* and *PG-BoostNoC* increases by 13.2% and 16%, respectively. However, the design footprint of the *LC* and the *RC* logic is a mere 1.77% of a single layered NoC, and hence, marginal when compared to the overall chip area.

8. CONCLUSION

In this paper, we demonstrate that the on-chip communication creates a severe performance bottleneck in many-core NTC systems. We demonstrate the key factors that affect the NoC performance at NTC, and propose *BoostNoC*—a novel power-efficient, multi-layered homogeneous NoC architecture that exploits the temporal variation in on-chip communication demand. *BoostNoC* efficiently switches between the two layers, *FruPuL* (optimized for power consumption), and *BoPeL* (optimized for performance), to extract the performance benefits of nominal voltage operation, while largely sustaining the energy benefits of NTC. We meticulously analyze the spatial and temporal communication patterns in *BoostNoC*, to further optimize by exploring two energy-efficient design derivatives: *PG BoostNoC* and *Drowsy BoostNoC*. Our analysis shows that *BoostNoC*, and its derivatives improve the energy-efficiency of a many-core NTC system by more than 35%.

References

1. R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. N. Mudge, Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits, *Proc. of the IEEE* **(2010)**, Vol. 98, pp. 253–266.

2. M. R. Seifi and M. Eshghi, Clustered noc, a suitable design for group communications in network on chip. *Journal on Computer and Electrical Engineering* 38, 82 **(2012)**.

3. A. Lankes, T. Wild, and A. Herkersdorf, Hierarchical nocs for optimized access to shared memory and io resources, *Euromicro Conference on Digital System Design (DSD)* **(2009)**, pp. 255–262.

4. C. Silvano, G. Palermo, S. Xydis, and I. S. Stamelakos, Voltage island management in near threshold manycore architectures to mitigate dark silicon, DATE, Dresden, Germany, March 24–28 **(2014)**, pp. 1–6.

5. M. Janidarmian, A. Khademzadeh, and M. Tavanpour, Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile-based network on chip. *IEICE Electronics Express* 6, 1 **(2009)**.

6. C. Rajamanikkam, R. J. Shridevi, S. Roy, and K. Chakraborty, Boostnoc: Power efficient network-on-chip architecture for near threshold computing, *IEEE International Conference on Computer-Aided Design (ICCAD)* **(2016)**.

7. 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. *IEEE Dependable Systems and Networks (DSN)* **(2012)**, pp. 1–11.

8. S. Mukhopadhyay, S. Ghosh, K. Kim, and K. Roy, Low-power and process variation tolerant memories in sub-90 nm technologies, *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)*, September **(2006)**, pp. 155–159.

9. U. R. Karpuzcu, A. A. Sinkar, N. S. Kim, and J. Torrellas, Energysmart: Toward energy-efficient manycores for nearthreshold computing, *Proceedings of High Performance Computer Architecture (HPCA)* **(2013)**, pp. 542–553.

10. E. Krimer, R. Pawlowski, M. Erez, and P. Chiang, Synctium: A near-threshold stream processor for energy-constrained parallel applications. *IEEE Computer Architecture Letters* 9, 21 **(2010)**.

11. B. C. Paul, A. Raychowdhury, and K. Roy, Device optimization for ultra-low power digital sub-threshold operation, *ACM International Symposium on Low Power Electronic Devices (ISLPED)* **(2004)**, pp. 96–101.

12. S. Mittal, A survey of architectural techniques for near-threshold computing. *J. Emerg. Technol. Comput. Syst.* 12, 46:1 **(2015)**.

13. M. Seok, D. Jeon, C. Chakrabarti, D. Blaauw, and D. Sylvester, Pipeline strategy for improving optimal energy efficiency in ultra-low voltage design, *DAC* **(2011)**, pp. 990–995.

14. N. R. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. N. Mudge, D. Sylvester, and D. Blaauw, Assessing the performance limits of parallelized near-threshold computing, *DAC* **(2012)**, pp. 1147–1152.

15. L. Chen and T. M. Pinkston, Nord: Node-router decoupling for effective power-gating of on-chip routers, *IEEE/ACM International Symposium on Microarchitecture (MICRO)* **(2012)**, pp. 270–281.

16. R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, Catnap: Energy proportional multiple network-on-chip, *International Symposium on Computer Architecture (ISCA)* **(2013)**, pp. 320–331.

17. H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, Run-time power gating of on-chip routers using look-ahead routing, *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, IEEE Computer Society Press **(2008)**, pp. 55–60.

18. H. Farrokhbakht, M. Taram, B. Khaleghi, and S. Hessabi, Toot: An efficient and scalable power-gating method for noc routers. *NOCS* **(2016)**, pp. 1–8.

19. A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, Energy-efficient interconnect via router parking, *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, IEEE Computer Society **(2013)**, pp. 508–519.

20. P. Bogdan and R. Marculescu, Workload characterization and its impact on multicore platform design, *2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS)* October **(2010)**, pp. 231–240.

21. P. Bogdan, M. Kas, R. Marculescu, and O. Mutlu, Quale: A quantum-leap inspired model for non-stationary analysis of noc traffic in chip multi-processors, *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, May **(2010)**, pp. 241–248.

22. I. Labs, The scc platform overview. May **(2010)**, Available: http://http://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-single-chip-platform-overview-paper.pdf.

23. K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, Drowsy caches: Simple techniques for reducing leakage power, *Proc. of 29th ISCA* **(2002)**, pp. 148–157.

24. T. E. Carlson, W. Heirman, and L. Eeckhout, Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation, *International Conference on Supercomputing* **(2011)**.

25. N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, A detailed and flexible cycle-accurate network-on-chip simulator, *IEEE International Symposium on Performance Analysis of Systems and Software* **(2013)**, pp. 86–96.

26. S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, The splash-2 programs: Characterization and methodological considerations, *International Symposium on Computer Architecture (ISCA)*, ACM **(1995)**, pp. 24–36.

27. L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch, Practical strategies for power-efficient computing technologies, *Proceedings of IEEE* (**2010**), Vol. 98, pp. 215–236.

28. D. Becker, Open source noc router rtl., August (**2012**). Available: https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router.

29. 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, *ACM/IEEE International Symposium on Networks-on-Chip (NOCS)* (**2012**), pp. 201–210.

30. W. Zhao and Y. Cao, Predictive technology model, June (**2012**). Available: http://ptm.asu.edu/.

31. S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, Varius: A model of process variation and resulting timing errors for microarchitects. *IEEE Transactions on Semiconductor Manufacturing* 21, 3 (**2008**).

32. H. Chen, D. Manzi, S. Roy, and K. Chakraborty, Opportunistic turbo execution in NTC: Exploiting the paradigm shift in performance bottlenecks, *IEEE/ACM Design Automation Conference (DAC)* (**2015**), pp. 63:1–63:6.

33. N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th edn., Addison-Wesley Publishing Company, USA (**2010**).

Chidhambaranathan Rajamanikkam

Chidhambaranathan Rajamanikkam *is a Ph.D. student at Utah State University, working in Bridge lab with Dr. Sanghamitra Roy and Dr. Koushik Chakraborty. Chidham's primary research interests include Hardware Security on 3PIPs and Neuromorphic Computing, Near-Threshold Computing, and Network-on-Chips (NoCs). In his recent research, he developed various detection techniques to determine the malicious activity in 3PIP cores. He is investigating various security implications emerge from neuromorphic computing system and investigating the interconnect performance in neuromorphic computing architectures.*

J. S. Rajesh

J. S. Rajesh *is currently pursuing the Ph.D. degree with the BRIDGE Laboratory, Electrical and Computer Engineering Department, Utah State University, Logan, UT, USA, under the mentorship of Dr. K. Chakraborty. He has also involved in third party IP security. In his earlier works, he explored energy efficient and fault tolerant router microarchitecture for the network-on-chip in the presence of supply voltage noise.*

Koushik Chakraborty

Koushik Chakraborty *is an Associate Professor in Electrical and Computer Engineering Department at Utah State University. He received the B.Tech degree from Indian Institute of Technology, Kanpur in 2000, and the M.S. and Ph.D. from the University of Wisconsin, Madison in 2004 and 2008, respectively. His current research interests are cross-layer circuit-architectural techniques to improve the energy efficiency and reliability of microprocessors. His works have won Best Paper Award at 2012 ICCD, and Best Paper Nominations at 2014 CODES-ISSS, 2011 DATE and 2010 VLSID, respectively. His research is currently funded by National Science Foundation and Micron Incorporation.*

Sanghamitra Roy

Sanghamitra Roy *is an Associate Professor in the department of Electrical and Computer Engineering at Utah State University. She received her Ph.D. degree in Electrical and Computer Engineering from the University of Wisconsin-Madison. Her doctoral research was sponsored by Intel Strategic CAD labs and National Science Foundation. She received her M.S. degree in Computer Engineering from Northwestern University in Dec 2003. Dr. Roy has authored over 50 peer reviewed publications in top tier journals and conferences as well as a book chapter in VLSI Design Automation. She serves in the editorial board of the IEEE Design and Test Magazine and in the technical program committees of DAC, ICCD and ISQED. She has won Best Paper Award nominations at the IEEE Design Automation and Test in Europe (DATE), 2011, IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2005, IEEE 23rd International Conference on VLSI Design (VLSI Design), 2010 and the Best Paper Award at the 30th IEEE International Conference on Computer Design (ICCD), 2012. She received the NSF CAREER Award in 2013. Her research interests are in VLSI circuit design and optimization and exploring reliability aware novel circuit styles and architectures.*