Improving DVFS in NoCs with Coherence Prediction

Robert Hesse Natalie Enright Jerger
Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto, Toronto, Canada
{hesserob, enright}@ece.utoronto.ca

ABSTRACT

As Networks-on-Chip (NoCs) continue to consume a large fraction of the total chip power budget, dynamic voltage and frequency scaling (DVFS) has evolved into an integral part of NoC designs. Efficient DVFS relies on accurate predictions of future network state. Most previous approaches are reactive and based on network-centric metrics, such as buffer occupation and channel utilization. However, we find that there is little correlation between those metrics and subsequent NoC traffic, which leads to suboptimal DVFS decisions. In this work, we propose to utilize highly predictable properties of cache-coherence communication to derive more specific and reliable NoC traffic predictions. A DVFS mechanism based on our traffic predictions, reduces power by 41% compared to a baseline without DVFS and by 21% on average when compared to a state-of-the-art DVFS implementation, while only degrading performance by 3%.

Categories and Subject Descriptors

C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures(Multiprocessors)—*Interconnection architectures*

General Terms

Design, Performance

Keywords

Networks-on-chip, dynamic voltage/frequency scaling, chip multiprocessor, cache coherence

1. INTRODUCTION

The steady increase in core counts coupled with power density limitations and the breakdown of Dennard scaling [10] demand NoCs that provide a scalable communication fabric for connecting a large number of resources within a chip. As the NoC expands to meet the performance demands of future many-core processors, it has also become a critical consumer of the CMP's overall power budget [11]. Dynamic Voltage and Frequency Scaling (DVFS) can limit the NoC's contribution to the chip's overall power envelope [16, 7, 5, 20].

NOCS'15 September 28-30, 2015, Vancouver, BC, Canada
 ISBN 978-1-4503-3396-2/15/09 ...$15.00
DOI: http://dx.doi.org/10.1145/2786572.2786595 .

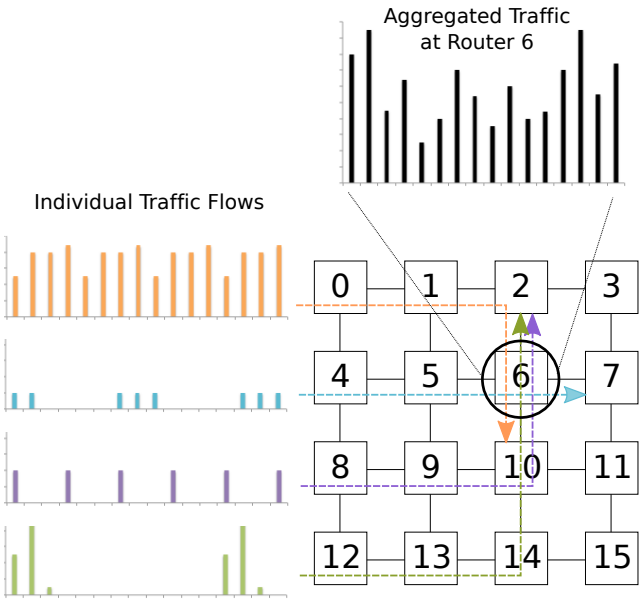

Figure 1: Aggregation of individual traffic flows

Although DVFS has been extensively studied in NoCs [16, 7, 5, 20], previous work mostly focuses on network-performance metrics, e.g., buffer occupation [7], channel utilization [20], roundtrip latency [6], or injection rate [14] to adjust the voltage/frequency (V/F) state of the NoC. Most of these approaches are *reactive* – they set V/F state based on currently observed network conditions and do not attempt to actively predict future communication requirements. For example, increased buffer occupation only triggers a reaction after the onset of congestion in the NoC and does not attempt to prevent congestion prior to its occurrence. These techniques assume that the future state of the network is predictable solely based on current network state, rather than application behaviour. This only works well when network traffic changes slowly and future network utilization correlates with its current state. In general, this is not the case for most real application traffic that exhibits abrupt changes and phase behaviour [1]. In Sec. 2.3, we show that no obvious correlation exists between current and future channel utilization. We find that predictions based purely on past and current network-centric metrics lead to high prediction errors.

Poor DVFS decisions can be very costly, either through performance loss due to *underpredicting* network utilization or wasted

energy caused by *overpredicting* NoC demands. Furthermore, it takes a long time to recover from poor DVFS decisions due to slow transition times for adjusting the voltage and frequency levels ($\sim$100s of cycles [7]). To avoid such costly mistakes, our work focuses on a novel, more reliable method to predict future NoC traffic based on the cause of the traffic, rather than its effect.

Traffic prediction in NoC is difficult [1] because every router in a NoC simultaneously transmits multiple traffic flows between many sources and destinations. Additionally, every source/destination pair exhibits its own dynamics. When we observe these individual dynamics in aggregate, it is almost impossible to detect predictable behaviour, since all traffic flows are interleaved, as seen in Fig. 1 at router 6. However, if instead of looking at the aggregate effect and trying to predict the future based on that, we look at the source of the individual traffic flows at routers 0, 4, 8, and 12, we can observe much more regular traffic patterns. We find that individual traffic flows are much more predictable than their aggregate.

To predict the behaviour of individual traffic flows, it is important to understand the underlying, dynamic causes of the traffic. For example, in NoCs for cache-coherent CMPs, the aggregate traffic consists of coherence protocol communication. In this work, we introduce a traffic prediction mechanism that relies on data collected from the coherence protocol, rather than the current state of the network. Recent work by Demetriades et al. [9] shows that coherence targets, defined as the subset of communication destinations for a particular thread, can be predicted with high probability by exploiting the inherent correlation between synchronization points in a program and coherence communication. We extend these findings and apply them to the prediction of bandwidth requirements to control DVFS in the NoC.

Our approach differs from existing DVFS techniques because it employs inherent application characteristics, rather than purely network-related metrics and it uses them to proactively predict DVFS-relevant properties of NoC traffic. This leads to a reduction in V/F mispredictions and more accurate DVFS decisions during runtime, which ultimately results in a more power-efficient NoC implementation without negatively impacting application throughput.

The primary contributions of this work are as follows:

- We show that current reactive DVFS mechanisms are not efficient at predicting future NoC demands.
- We develop a reliable traffic prediction mechanism that is 87% accurate in predicting the bandwidth requirements based on application cache coherence behaviour.
- We propose a low-overhead hardware implementation of our predictive DVFS mechanism for router-based voltage frequency islands (VFI), which improves power-delay product by 39% over a baseline with static V/F levels and by 21% over prior work.

2. BACKGROUND AND MOTIVATION

First, we discuss the basics of cache coherent NoCs and dynamic voltage and frequency scaling (DVFS). Next, we cover related work in DVFS for NoCs. Finally, we motivate our novel DVFS prediction mechanism.

2.1 Background

Cache-Coherent NoCs In this work, the NoC provides the communication infrastructure for a shared-memory CMP that relies on a directory-based cache coherence protocol. Here, NoC traffic consists of messages defined by the coherence protocol (e.g., MESI, MOESI), such as data requests and replies, acknowledgements (e.g., ACK, NACK), and invalidation requests, which are sent between communication end points. Communication end points are cache controllers, directory controllers, and memory controllers. In general, NoCs are oblivious to the type of traffic they transfer; we exploit properties of cache coherence traffic. Specifically, we use the predictability of destination sets [15], which are defined by the collection of processors that receive an individual coherence request. Therefore, our work is fundamentally tied to cache-coherent NoCs.

DVFS Dynamic voltage and frequency scaling can adaptively decrease the supply voltage by reducing the switching frequency. This can achieve significant power and energy savings (energy consumption is reduced quadratically with the decrease of voltage), but may incur various performance penalties due to lower switching frequency. Therefore, DVFS must carefully trade off power and energy consumption with performance.

Voltage and frequency islands (VFIs) [18] allow for fine-grained adjustment of voltage and frequency (V/F) levels throughout the NoC. Coupled with globally asynchronous, locally synchronous design in which a CMP is divided into individual tiles operating with their own clock and voltage domains [17], designers can exploit both temporal and spatial variation in NoC traffic to reduce power consumption during runtime. Recent developments in on-chip voltage regulators [21] are key enablers for fine-grained VFIs.

2.2 Related Work

We focus our discussion of related work on DVFS mechanisms for NoCs and coherence prediction.

DVFS Work on DVFS for NoCs can generally be divided into single V/F domain scenarios and more fine-grained designs based on multiple VFIs. Policies for a single V/F domain usually measure some network performance metric, such as injection rate [14] or Average Memory Access Time (AMAT) [6] and adjust the entire NoC's voltage reactively. Won et al. [23] rely on Artificial Neural Networks to predict program phase patterns and their influence on NoC traffic. Their proactive DVFS approach allows for additional energy savings without degrading performance. Our work also focuses on proactive DVFS policies, but applies them to VFI-based DVFS instead of a single V/F domain, because it allows us to exploit more fine-grained spatial and temporal traffic variations. While VFI-based DVFS incurs more interfacing overhead [14], the goal of our work is to mitigate the additional overhead by increasing DVFS efficiency.

VFI-based DVFS policies include dynamic voltage scaling of individual links [20] and an approach based on a fractional state model [5]. DVFS can be applied by monitoring queue occupancy [7] on the Intel SCC [11] or using a combination of link utilization and buffer occupancy [19]. Mishra et al. [16] use the buffer occupancy of neighbouring routers to adjust the V/F level of upstream routers. These proposals rely on purely network-related metrics and use a reactive DVFS approach. Our work differs from prior work in that it makes proactive predictions about the future NoC state and that it relies on coherence communication, as an application-centric metric, rather than purely network-based metrics.

Coherence Prediction Cache misses in directory-based coherence protocols rely on indirections through the directory for cache-to-cache communication, which incurs additional latency. Destination set predictors can improve miss handling latency by predicting the destination of a coherence request instead of a costly indirection to the directory [15]. Demetriades et al. [9] improve the prediction accuracy of destination set predictors by exploiting the inherent correlation between synchronization points in a program and coherence communication. Our work is based on their findings and applies it to DVFS for cache-coherent NoCs.

Coherence property	Derived NoC property	Possible action
Destination set (Destinations and corresponding volume of messages)	Bandwidth requirements (Exact location and volume of traffic)	DVFS
		Power gating
	Traffic flows	Set up forwarding paths, adaptive routing
	Hotspot prediction	Adapt routing, throttle injection
	Buffer requirements	Dynamic buffer allocation
Type and mix of coherence messages	Criticality/priority	Adjust QoS
	Volume and burstiness of traffic	Resource allocation (bandwidth, circuits)
	Average packet size	Buffer requirements
	Are responses expected?	Set up circuit/return path

Table 1: Relation between coherence communication properties and their NoC implications

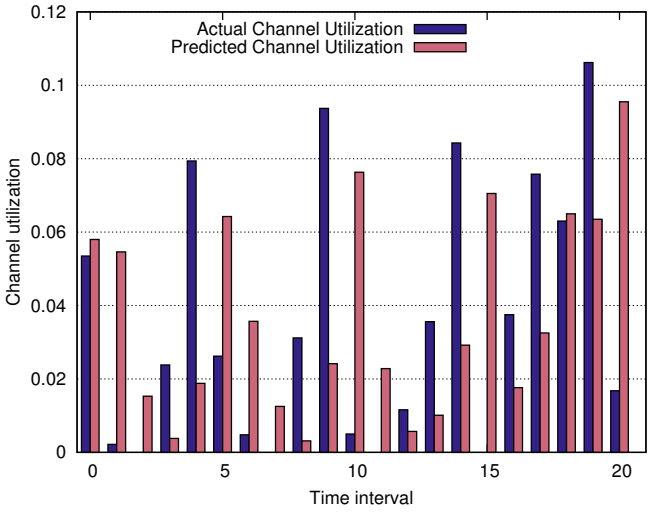

Figure 2: Channel utilization prediction

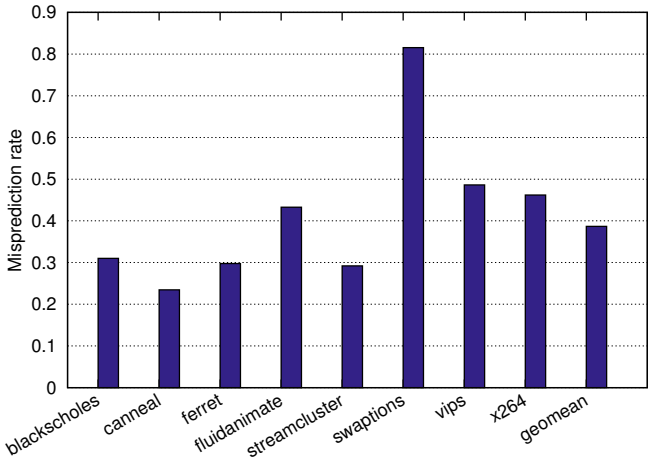

Figure 3: Prediction error for channel utilization in PARSEC benchmarks

2.3 Motivation

In this section, we first show how current DVFS solutions fail to predict future traffic demands and explore why traffic prediction in NoCs is a difficult problem. Next, we show how properties of the cache coherence protocol can improve prediction accuracy.

Although recent DVFS approaches improve NoC energy efficiency by adapting network bandwidth based on spatial and temporal variations in traffic requirements [5, 20], they are mostly reactive and focus on general network-related metrics and thus do not immediately take changes in application behaviour into consideration. For good power and performance results, DVFS needs to be aware of an application's network demands ahead of time and needs to be tuned to fluctuating network bandwidth requirements based on the actual application's requirements and not a proxy value that may not correlate well.

Common proxies include link utilization [20], buffer occupation [7], injection bandwidth [14], and round-trip delay [6]. Based on the individual metric's past behaviour in a local network context, the voltage and frequency of either individual links [20] or entire routers [16] are adjusted. However, we do not find a strong, intuitive correlation between these metrics and the actual future bandwidth demands. For example, simply because a link has been utilized heavily in the past, does not guarantee that it will show the same behaviour in the immediate future. This is especially true for sporadic changes that occur with the phase-based communication behaviour seen in CMP workloads [1]. To demonstrate this mismatch, we pick a random interval from *fluidanimate* executed on a 4×4 mesh NoC, and show one link's actual utilization compared to its predicted utilization in Fig. 2. The particular prediction algorithm used in this experiment [20] results in a prediction error of 270% because there is no obvious correlation between past and future behaviour. We measure the prediction error to be 39% on average across a selection of PARSEC [3] benchmarks, as shown in Fig. 3. Prediction mechanisms that focus on these types of network-related metrics have limited accuracy; DVFS mechanisms that rely on their predictions perform poorly. Consequently, there exists a need for more accurate traffic prediction to improve NoCs power efficiency using DVFS.

NoC traffic prediction is difficult; the NoC is a shared communication medium which transmits the traffic of several sources simultaneously. This leads to the overlap of individual traffic flows at each router. Additionally, each thread running on a different core may be in a different state and therefore exhibit changes in communication behaviour. Each node experiences interleaved traffic behaviour from different threads at different times. We need to understand the origin of the traffic to make proper assumptions about their behaviour and their individual contributions to the entirety of the NoC's traffic. Once we are able to understand and predict the communication behaviour of each individual thread at its source, we can derive their contributions to the local context of a router or link and thus deduce a global picture of the upcoming bandwidth requirements.

In cache-coherent NoCs, the traffic at each source consists of

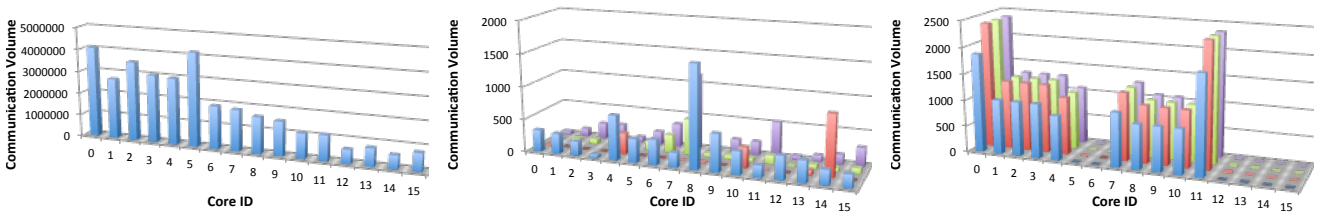

Figure 4: Communication distribution of Core 0 in fluidanimate: (a) as seen during the whole execution. (b) as seen during the execution of four consecutive fixed-time sub-intervals. (c) as seen across four different dynamic instances of the same sync-defined interval

messages defined by the coherence protocol, i.e., data requests and replies, invalidations, ACKs, sent between cache, directory, and memory controllers. In Table 1, we take a detailed look at how certain properties of cache coherence traffic can influence NoC metrics and be exploited by the NoC. For example, certain types of coherence messages are followed by a direct response; NoC performance could be improved by anticipating this response and setting up an optimized return path. While many interesting NoC-related properties can be derived from coherence protocol messages, we are mainly interested in temporal and spatial variations of bandwidth requirements; thus we limit the scope of our observations to the volume of data communication between source/destination pairs over time, which is defined by the amount of coherence messages exchanged. Since coherence messages generally follow more regular and predictable patterns [15] than aggregate bandwidth in the network, we use them to derive more accurate predictions about upcoming NoC bandwidth requirements.

Fig. 4 demonstrates how we can use synchronization points and destination sets to expose more predictable traffic patterns. Fig. 4a shows the communication volume between core 0 and all other cores over the entire runtime. If we analyze the communication volume per destination for fixed time intervals (Fig. 4b), distinct predictable features that could be used to predict core 0's bandwidth contribution do not emerge. Recent work by Demetriades et al. [9] uses an application's synchronization points as interval boundaries, rather than predetermined, fixed time intervals. Synchronization points, such as locks, barriers, joins, etc. indicate points when certain data private to a processor will become visible–and possibly be communicated–to other processors. Synchronization points likely indicate behaviour change and thus the data communication following different instances of the same synchronization point should be more predictable and have greater repeatability. In Fig. 4c, we demonstrate how the granularity of synchronization-based intervals exposes repeatable and predictable traffic patterns when communication volume is tracked per synchronization point.

Synchronization-based intervals allow us to create highly accurate destination set predictors, which we can use to determine temporal and spatial bandwidth requirements throughout the NoC to trigger informed DVFS decisions. To the best of our knowledge, this work is the first to predict NoC-related metrics derived from the behaviour of cache coherence communication.

3. IMPLEMENTATION

In this section, we describe how DVFS based on coherence prediction can be implemented in hardware and detail the prediction algorithm.

Our design is presented in the context of a Global Asynchronous Local Synchronous (GALS) 2D mesh NoC design [17]. In this implementation, we assume separate voltage-frequency islands for each router.[1] Fig. 5 shows an overview of the architecture. In addition to an existing router-based DVFS mechanism, the architecture adds two modules at each tile: a Coherence Prediction Engine (CPE), co-located with the cache and directory controllers, and an Accumulation and Decision Module (ADM) in every router. The basic idea is that CPEs predict future NoC communication requirements based on coherence behaviour and communicate those predictions through the NoC. ADMs in the routers then collect this data and evaluate it to make appropriate changes to the voltage and frequency levels.

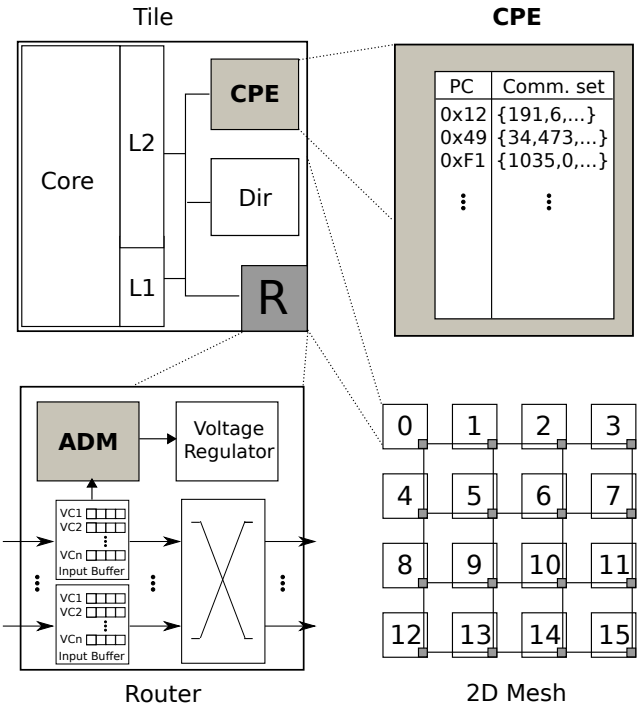

Figure 5: Overview of the architecture.

3.1 Coherence Prediction Engine (CPE)

The CPE is the main architectural component of our implementation. It is designed to accurately predict future NoC communication based on current and previously experienced coherence behaviour and then send that data to relevant ADMs. To extract relevant data from the coherence protocol, the CPE is connected to the coherence controllers (cache and directory) at its tile and monitors their traffic. It records synchronization epochs (sync-epochs) along with the

[1]Implementations with different granularities of VFIs [18] are possible with only minor adjustments.

corresponding communication set (traffic volume per destination) and later predicts NoC traffic based on those recordings.

Sync-Epoch Based Prediction As briefly motivated in Sec. 2.3 and described in more detail by Demetriades et al. [9], synchronization-points (sync-points) can be used for accurate communication set prediction. A sync-point is an execution point at which a software synchronization routine is invoked (e.g., barrier, join, wakeup, broadcast, lock, unlock). Each sync-point can be identified by its type, its static calling location (program counter), and a dynamic ID, which expresses multiple dynamic instances of the same static sync-point. Sync-epochs define the execution time between two consecutive sync-points. Therefore, each sync-point marks the end of one sync-epoch and the beginning of the next. Sync-points are executed repeatedly and create a sequence of dynamic instances for each sync-epoch. As these instances exercise the same or similar code and operate on the same (or related) data structures, it is likely that there are behavioural similarities between them [8]. Such similarities may also be reflected in the communication behaviour. We use this property to recall past communication patterns at a sync-epoch granularity to predict future communication sets. Sync-epoch based prediction requires synchronization primitives to be exposed to the hardware.

Prediction Table A prediction table keeps track of past communication sets. During the execution of a sync-epoch, communication counters monitor the spatial distribution of the traffic that originates at a particular node. At the end of a sync-epoch, the current communication set is stored in the prediction table and a prediction is made for the next sync-epoch. For this prediction, previously stored communication sets for the current static sync-point are retrieved from the prediction table. Each table entry holds a sequence of communication sets for a single static synch epoch. Each communication set holds N w-bit integers, representing the amount of traffic to the N possible destinations in a NoC with N nodes. w is a design parameter that we chose as 8. Empirically, the amount of traffic to a single destination during a sync-epoch never exceeded 100,000 flits, so a 17-bit integer is able to hold this information. At the same time, we determined that small values (less than 512) are insignificant for our purposes, so we truncate the 17 bits down to 8 to minimize storage overhead. Table entries are indexed with the program counter of the sync-epoch. Since none of the PARSEC benchmark includes more than 64 static sync-points and prior work [9] suggests a history depth of 2 as sufficient for most traffic patterns, we design the prediction table with 128 entries. Therefore, the storage requirement for a single prediction table is $128 \times w \times N = 2KB$ for a 16-node NoC.

Once a prediction has been retrieved from the prediction table, the CPE sends single-flit control packets containing the 8-bit traffic volume prediction for the new sync-epoch to each of the destinations recorded in the communication set. These control flits are injected into the NoC alongside regular data packets and do not require a separate control network. The overhead caused by control flits is minimal ($\sim 0.2\%$) due to the low frequency of sync-points.

3.2 Accumulation and Decision Module (ADM)

The ADM (Fig. 5) collects the bandwidth prediction from those control flits that pass through a specific router. Per-destination control flits traverse the same route through the NoC as their corresponding traffic flow when using a deterministic routing algorithm such as dimension-order routing. These control flits carry the impending bandwidth requirements between a source-destination pair during the upcoming sync-epoch. The ADM extracts the bandwidth prediction from all control flits that pass through and accumulates them to gauge the impending aggregate bandwidth requirements for all relevant traffic flows.

Each ADM keeps track of all bandwidth requirements in aggregate, instead of storing individual traffic flow bandwidths for each source-destination pair. This allows us to reduce the storage requirements per router significantly by only saving its aggregate bandwidth prediction in a single register. Control flits passing through an ADM indicate a change in bandwidth requirements for a specific source-destination pair by containing a delta value (difference between its previous prediction and its current prediction), which is used to update the ADM's aggregate bandwidth prediction. For example, let us assume that router 6's ADM currently holds an aggregate bandwidth prediction of 20000 flits in its register, when a new control flit from router 0 on its way to router 10 passes through. It carries a value of -4000, because the communication set prediction at router 0 predicted 4000 fewer flits will be sent to router 10 during the current sync-epoch compared to the last sync-epoch, which will update the aggregate bandwidth prediction at router 6 to 16000 flits.

Based on the aggregate BW prediction, the ADM makes a decision regarding the V/F level of its router and communicates this to the voltage regulators. We use empirically determined static thresholds to map bandwidth ranges to V/F levels (Table 3). Decisions are made whenever a new control flit arrives–after the aggregate BW has been updated. Initially, we enforced minimum intervals between V/F updates as a safety measure against too frequent updates to limit transition overheads. However, we found them to be unnecessary due to sufficiently spaced control flits (millions of cycles).

3.3 DVFS mechanism

Our implementation uses per-core on-chip DVFS, where each router represents a different VFI. We adopt the two-step voltage regulator configuration as proposed by Kim et al. [13]. An off-chip regulator performs the initial step down from the supply voltage to 1.8V, followed by multiple on-chip voltage regulators. The on-chip regulators operate at 125MHz switching frequency and provide voltage transitions between 1.25V to 0.8V. This allows for router frequencies ranging from 1.0GHz to 2.25GHz. The routers can operate at the lower frequency during frequency step-down by quickly ramping down the frequency before the voltage steps down. For stepping-up the frequency using DVFS, we first step-up the voltage before ramping up the router frequency. The overhead in every transition is primarily the voltage settling time which is 13ns for every 100mV change [13]. Therefore, due to higher than required voltage during step-down (and lower frequency during step-up), the power consumed by the router during a transition lies between the values before and after the scaling. All our evaluations take this overhead into account. It is important to note that the entire mechanism, including CPE and ADM, works asynchronously and can therefore be implemented off the critical path.

3.4 Discussion

Here, we discuss some of the implementation details that are not directly related to the architecture.

VFI Granularity Our particular implementation relies on router-based voltage frequency islands, however coherence prediction can also be used for different granularities of VFIs with minor modifications. It is equally applicable to cluster or region-based DVFS [18] by using one ADM per cluster, instead of one per router. In this case, the ADM would keep track of cluster-based bandwidth requirements. Alternatively, by using one ADM per link, very fine-grained DVFS could be implemented.

Topology and Routing The way bandwidth predictions are com-

# of Cores/Threads	16/16, 1GHz
L1 Cache (D & I)	private, 4-way, 32KB each, 64 Byte Blocks
L2 Cache	shared, distributed, 8-way, 512KB each
Cache Coherence	MOESI distributed directory
Router	wormhole, VC, 3 stages, 2GHz, DOR
VCs/Buffer Depth	4/4 Flit, 8 Bytes
Control Interval	100,000 cycles at 1GHz

Table 2: Simulation parameters

Voltage level [V]	0.8	0.85	0.9	1.0	1.1	1.2
Frequency level [GHz]	1.0	1.25	1.5	1.8	2.0	2.25
BW range [10^4 flits]	<1	1-2	2-4	4-7	7-8	>8

Table 3: DVFS parameters and mapping from ADM's aggregated BW prediction to corresponding V/F levels

municated through the NoC and accumulated at the routers, currently relies on a deterministic routing algorithm. We require that control flits take the same path as other packets belonging to a specific traffic flow. Adjusting the implementation for adaptive routing algorithms is possible, but would be slightly more complicated. Bandwidth estimates per path would need to be probabilistic and account for path diversity. Control flits would need to be sent along each possible path between two nodes.

Optimizations We consider several optimizations to our base implementation, which can be used to trade off implementation cost and prediction accuracy, as well as bandwidth overhead. For example, the number of control flits sent can be reduced by omitting minor updates to the bandwidth prediction and instead bundling multiple minor updates into a single, less frequent control flit. Another optimization addresses the storage overhead of both CPEs and ADMs. Instead of keeping track of exact bandwidth numbers, coarse bandwidth categories (e.g., low, medium, high) could be defined. Prediction tables and ADMs would then store and operate on these less-accurate categories. We evaluate this optimization in Sec. 4.

Lack of Sync-Points and Fallback If no history exists for a particular sync-epoch, or a substantial mismatch between predicted bandwidth and actual bandwidth is detected, the CPE performs a new prediction based on recent communication history and ADMs are updated accordingly. As a fallback to prevent severe performance penalties in the case of mispredictions, routers additionally monitor their buffer occupancy and can override the bandwidth prediction if occupancy levels exceed a predetermined threshold.

4. EVALUATION

In this section we describe our experimental setup and subsequently evaluate our proposed prediction mechanism. Our baseline platform is a 16-core CMP with a 2-level cache hierarchy, split, private L1 caches, and a distributed, shared L2 last-level cache. Cache coherence is maintained via a MOESI directory cache coherence protocol. The NoC topology is a 4×4 2D mesh, with each router attached to a single processor core. Table 2 summarizes the baseline CMP setup. Simulation experiments are performed using the gem5 [4] full system simulator, with the Ruby memory model and a modified version of BookSim [12] for cycle-accurate NoC simulation. The benchmark applications are taken from the PARSEC benchmark suite [3]. Each application is executed for 100 million cycles within its region of interest (ROI), using the *simmedium* input set. We use DSENT [22] to model delay, static and dynamic power for a 22nm process. Additionally, we implemented an RTL

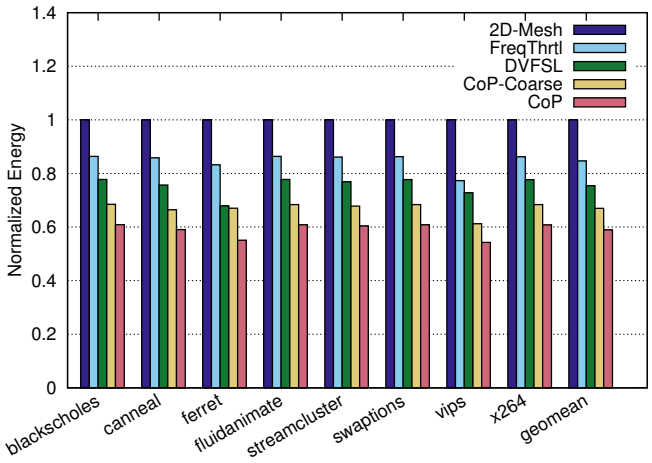

Figure 6: Power results

model of our mechanism on top of the standard BookSim router RTL implementation by Becker [2]. Synthesis using Synopsys Design Compiler with a TSMC 65nm standard library confirms the DSENT model and results in 0.3% area overhead per tile.

To focus on the evaluation of the NoC DVFS mechanisms, the core frequency is fixed at 1GHz throughout the simulations. We use 6 frequency levels between f_{max} = 2.25GHz and f_{min}= 1.0GHz for the NoC (see Table 3). For each frequency, there is a corresponding voltage level between 1.2V and 0.8V, which is roughly the minimum voltage allowing correct operation. We assume that each step V/F level change takes 100 core cycles (100 cycles per step is sufficient assuming on-die regulation [21]). During V/F transitions, the router operation is halted. All V/F related parameters are inspired by the Intel SCC [11] DVFS implementation.

We compare the following 5 mechanisms; the last two are our proposals in this work:

- **2D Mesh-2GHz:** The baseline mesh constantly operates at 2GHz
- **FreqThrtl:** Proposed by Mishra et al. [16] as a reactive mechanism that adjusts V/F according to local congestion. Routers operate at 0.8*f_{base} (f_{base}=2.0GHz) without congestion and faster in a congested state.
- **DVFSL:** Proactive DVFS mechanism, proposed by Shang et al. [20] that relies on buffer occupancy and channel utilization to adjust link V/F. Instead of link V/F, we use their algorithm to adjust router V/F to provide better comparability.
- **CoP-Coarse:** Coherence Prediction using sync-epochs and bandwidth categories instead of exact bandwidth estimates.
- **CoP:** Coherence Prediction using sync-epochs with exact bandwidth predictions

We chose *FreqThrtl* and *DVFSL*, because both are mechanisms that use VFI at a router granularity–one represents a reactive DVFS mechanism (*FreqThrtl*) and the other is predictive (*DVFSL*).

Normalized power results are displayed in Fig. 6. Naturally, the baseline uses the most power, because it cannot scale down its voltage or frequency. Compared to the baseline, our best mechanism can reduce power consumption by 41%, while the performance only degrades by 3% (Fig. 7), which leads to a power-delay reduction of 39% (Fig. 8). Both, *FreqThrtl* and *DVFSL* perform better than the baseline in terms of power consumption, but worse than any of our proposed mechanisms.

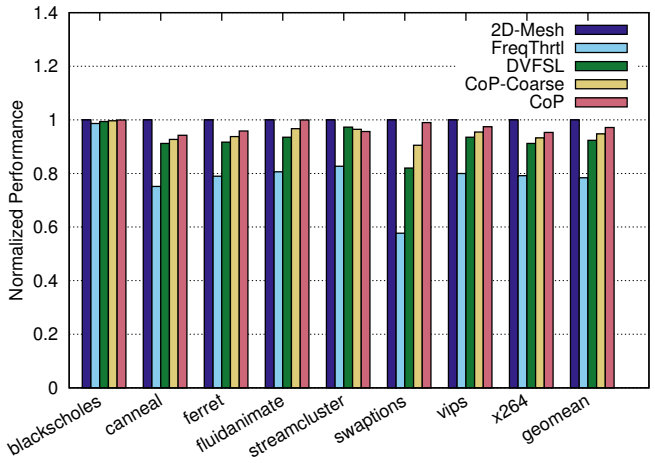

Figure 7: Performance results

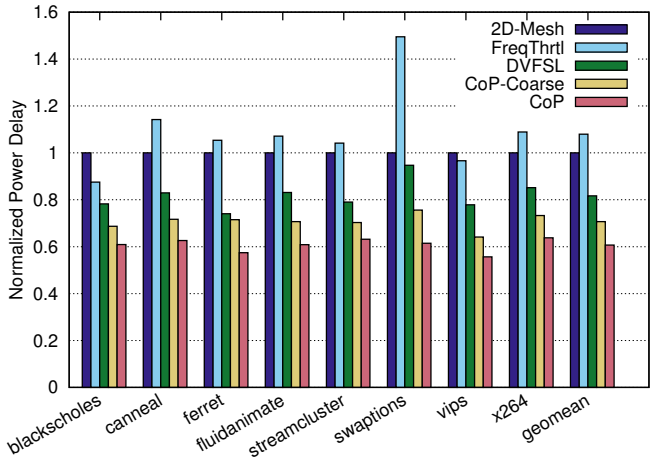

Figure 8: Power delay results

FreqThrtl suffers from the fact that buffer utilization is generally low for PARSEC benchmarks and only rarely exceeds its congestion threshold to increase the frequency. Hence, all routers operate at 1.5GHz (0.8*f_{base}) for most of the time, which leads to 17% power savings, but at the same time performance drops significantly due to the slow routers. We tried to adjust *FreqThrtl's* buffer utilization thresholds to gain a more even distribution of frequency states, but the low buffer utilization prevented us from finding a suitable setting. Depending on the chosen threshold, either all routers were running at f_{base}, or all routers were running at 0.8* f_{base} with little variation in between.

DVFSL performs better than *FreqThrtl*, but is negatively affected by a high prediction error rate, which result in non-optimal V/F choices. Fig. 9 shows that *DVFSL's* mean average percentage error (MAPE) for predicting buffer occupancy and channel utilization is 39%. *CoP*, on the other hand, is able to predict the bandwidth requirements per router with 87% accuracy on average. The high prediction accuracy is the main reason for making correct and beneficial DVFS decisions. *CoP-Coarse* trades off prediction accuracy (26% error rate) for smaller prediction tables, which leads to 8% worse power consumption and 3% lower performance. At the same time, it reduces storage overhead for prediction tables inside the ADMs by 75% to 512B each. These results emphasize the importance of accurate, proactive DVFS decisions in order to save significant amounts of energy while not degrading performance.

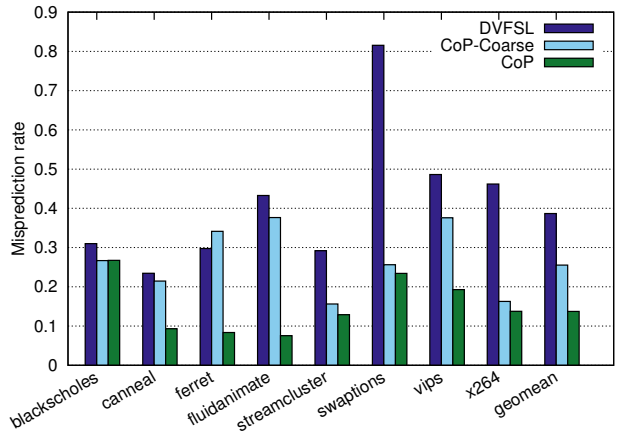

Figure 9: Prediction error

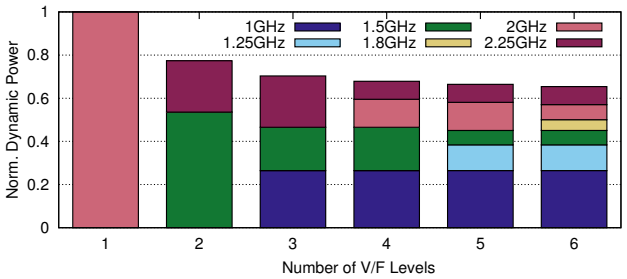

Figure 10: Distribution of dynamic power consumption between different V/F levels (averaged across PARSEC benchmarks)

Design space exploration To provide more insight into the choice of some of our parameters, we performed a design space exploration for the amount of different V/F levels provided by the DVFS mechanism, and the history depth of the CPE's prediction table. In Fig. 10 we measure the reduction of dynamic power consumption when the number of V/F levels is increased for *CoP*. The results show each V/F level's contribution to the total dynamic power consumption, as well as how much additional power can be saved by adding another V/F level. We found that providing more than 6 V/F levels leads to diminishing returns. From Fig. 11 it is evident that a prediction table history depth of more than 2 also leads to diminishing returns for the prediction error. While costly in terms of area and power consumption, adding a second entry for the history depth significantly reduces the prediction error by 15% due to the added ability to recognize alternating patterns across sync-epoch instances, which occur relatively frequently.

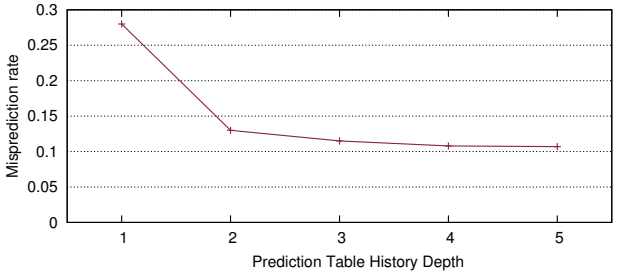

Figure 11: Effect of prediction table history depth on prediction accuracy

5. CONCLUSION

The NoC constitutes a significant and increasing part of overall CMP power consumption. This work focused on a novel prediction mechanism for improved VFI-based DVFS in order to reduce power consumption while maintaining performance. Instead of predicting bandwidth requirements using unrelated metrics or in aggregate throughout the network, this work uses sync-epoch based cache coherence prediction to reliably predict individual traffic flows. The individual traffic flows are then used to infer aggregate bandwidth demands in the network that enable informed DVFS decisions. A high prediction accuracy of 87% leads to an improved power-delay product of 39%.

Acknowledgements

The authors thank the anonymous reviewers for their thorough suggestions on improving this work. This work is supported by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, the Ministry of Research and Innovation Early Researcher Award and the University of Toronto.

6. REFERENCES

[1] M. Badr and N. Enright Jerger. SynFull: Synthetic traffic models capturing cache coherent behaviour. In *International Symposium on Computer Architecture (ISCA)*, pages 109–120, June 2014.

[2] D. U. Becker. *Efficient Microarchitecture for Network-on-Chip Routers*. PhD thesis, Stanford University, August 2012.

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proc. of PACT*, Oct. 2008.

[4] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[5] P. Bogdan, R. Marculescu, S. Jain, and R. Gavila. An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads. In *International Symposium on Networks on Chip (NoCS)*, pages 35–42, May 2012.

[6] X. Chen, Z. Xu, H. Kim, P. Gratz, J. Hu, M. Kishinevsky, and U. Ogras. In-network monitoring and control policy for DVFS of CMP networks-on-chip and last level caches. In *International Symposium on Networks on Chip (NoCS)*, pages 43–50, May 2012.

[7] R. David, P. Bogdan, and R. Marculescu. Dynamic power management for multicores: Case study using the Intel SCC. In *Internationa Conference on VLSI and System-on-Chip (VLSI-SoC)*, pages 147–152, Oct 2012.

[8] S. Demetriades and S. Cho. BarrierWatch: Characterizing multithreaded workloads across and within program-defined epochs. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, CF '11, pages 5:1–5:11, 2011.

[9] S. Demetriades and S. Cho. Predicting coherence communication by tracking synchronization points at run time. In *Proceedings of the International Symposium on Microarchitecture*, MICRO-45, pages 351–362, 2012.

[10] R. Dennard, V. Rideout, E. Bassous, and A. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, Oct 1974.

[11] J. Howard, S. Dighe, S. R. 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. K. De, and R. Van Der Wijngaart. A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. *IEEE Journal of Solid-State Circuits*, 46(1), January 2011.

[12] 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. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 86–96, April 2013.

[13] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *International Symposium on High Performance Computer Architecture*, pages 123–134, Feb 2008.

[14] G. Liang and A. Jantsch. Adaptive power management for the on-chip communication network. In *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, pages 649–656, 2006.

[15] M. Martin, P. Harper, D. Sorin, M. Hill, and D. Wood. Using destination-set prediction to improve the latency/bandwidth tradeoff in shared-memory multiprocessors. In *International Symposium on Computer Architecture*, pages 206–217, June 2003.

[16] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das. A case for dynamic frequency tuning in on-chip networks. In *Proceedings of the International Symposium on Microarchitecture*, MICRO 42, pages 292–303, 2009.

[17] J. Muttersbach, T. Villiger, and W. Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 52–59, 2000.

[18] U. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung. Design and management of voltage-frequency island partitioned networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(3):330–341, March 2009.

[19] A. Rahimi, M. E. Salehi, S. Mohammadi, and S. M. Fakhraie. Low-energy GALS NoC with FIFO-monitoring dynamic voltage scaling. *Microelectronics Journal*, 42(6):889 – 896, 2011.

[20] L. Shang, L.-S. Peh, and N. Jha. Power-efficient interconnection networks: Dynamic voltage scaling with links. *Computer Architecture Letters*, 1(1):6–6, January 2002.

[21] A. Sinkar, H. Ghasemi, M. Schulte, U. Karpuzcu, and N. S. Kim. Low-cost per-core voltage domain support for power-constrained high-performance processors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(4):747–758, April 2014.

[22] 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 *Proc. of the International Symposium on Networks-on-Chip*, May 2012.

[23] J.-Y. Won, X. Chen, P. Gratz, J. Hu, and V. Soteriou. Up by their bootstraps: Online learning in artificial neural networks for CMP uncore power management. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 308–319, Feb 2014.