DVFS for NoCs in CMPs: A Thread Voting Approach

Yuan Yao and Zhonghai Lu
KTH Royal Institute of Technology, Stockholm, Sweden
{yuanyao, zhonghai}@kth.se

ABSTRACT

As the core count grows rapidly, dynamic voltage/frequency scaling (DVFS) in networks-on-chip (NoCs) becomes critical in optimizing energy efficacy in chip multiprocessors (CMPs). Previously proposed techniques often exploit inherent network-level metrics to do so. However, such network metrics may contradictorily reflect application's performance need, leading to power over/under provisioning.

We propose a novel on-chip DVFS technique for NoCs that is able to adjust per-region V/F level according to voted V/F levels of communicating threads. Each region is composed of a few adjacent routers sharing the same V/F level. With a voting-based approach, threads seek to influence the DVFS decisions independently by voting for a preferred V/F level that best suits their own performance interest according to their runtime profiled message generation rate and data sharing characteristics. The vote expressed in a few bits is then carried in the packet header and spread to the routers on the packet route. The final DVFS decision is made democratically by a region DVFS controller based on the majority election result of collected votes from all active threads. To achieve scalable V/F adjustment, each region works independently, and the voting-based V/F tuning forms a distributed decision making process.

We evaluate our technique with detailed simulations of a 64-core CMP running a variety of multi-threaded PARSEC benchmarks. Compared with a network without DVFS and a network metric (router buffer occupancy) based approach, experimental results show that our voting based DVFS mechanism improves the network energy efficacy measured in MPPJ (million packets per joule) by about 17.9% and 9.7% on average, respectively, and the system energy efficacy measured in MIPJ (million instructions per joule) by about 26.3% and 17.1% on average, respectively.

1. INTRODUCTION

Network-on-Chip (NoC) is becoming the de-facto mainstream interconnect scheme for today's and future high performance CMPs and MPSoCs. As multi-core computers enjoy consistently performance enhancement with a growing number of cores, energy/power consumption has emerged as the main constraint for current chip design. According to the studies in [7, 11, 17], on-chip networks can consume a substantial fraction (which may potentially amount to 30-40%) of the entire chip power.

An important direction towards increasing the energy efficacy of NoC in multi- or many-core systems has focused on Dynamic Voltage/Frequency Scaling (DVFS). A DVFS mechanism is desired to co-optimize system-level performance and power, i.e., to obtain as much as performance by spending as less power as possible. Conventionally, DVFS mechanisms in NoC have been designed [9, 23, 3, 20] to utilize network-level performance metrics, such as network latency, router buffer load or network throughput. These metrics capture inherent performance characteristics of the network itself, but are often insufficient to reflect application-level or system-level performance. Specifically, network delay may not be indicative of network-related stall-time at the processing core [10]. This is because much of the packets' latency can be hidden by various latency-hiding techniques such as read/write buffers. The router buffer load might not accurately reflect the data criticality to system performance because it has no knowledge of data dependency among different threads. Given that the buffer-load is low, it may not always be proper to lower the on-chip voltage/frequency (V/F) level since the completion time of one request may affect the progress of other data-sharer cores. Relying on network throughput might not be effective in V/F tuning since CMPs are self-throttling: a core cannot inject new requests into the network once it fills up all of its miss request buffers. A low throughput can either indicate an under-saturated network or an over-saturated network when the request buffers of cores are fully filled. Because of the above reasons, NoC DVFS mechanisms relying merely on network-level metrics may improperly adjust on-chip V/F levels, leading to over- and under-provisioning of power and thus inferior energy efficacy.

In the paper, we propose a novel on-chip DVFS strategy that is able to adjust network V/F levels according to a thread's direct performance indicatives in terms of *message generation rate* and *data sharing* metrics. Like the conventional network metric based approaches, the injection rate allows to capture a thread's communication need. The data sharing characteristics in a cache-coherent multicore enable to take data criticality and dependency into consideration. In this way, our strategy can effectively overcome the power over- and under-provisioning problems whenever present. In order to achieve a light-weight solution, our technique allows a thread to *vote* a preferred V/F level according to its own performance indicatives, and the V/F votes (a few bits) are carried by packets and spread into the routers. A region-

978-1-4673-9211-2/16/$31.00

based V/F controller collects all votes of passing packets and makes a *democratic* decision on the final V/F level for its region composed of a few routers. As such, we call our technique a *thread voting* approach, which means a thread's active participation in influencing the V/F decisions, in sharp contrast to existing thread data-sharing oblivious V/F adjustment techniques.

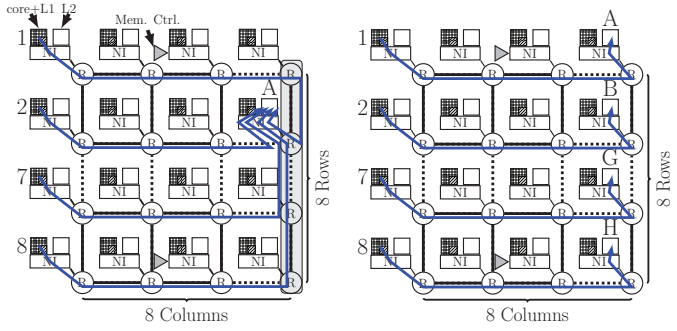
Under a multi-threaded programming environment, we evaluate our DVFS mechanism with cycle-accurate full-system simulations in Gem5 [5] using PARSEC benchmarks [4]. We further exploit McPAT [18] for reporting power related results. In the benchmark experiments, we expose the power over- and under-provisioning problem of a purely network-metric based DVFS mechanism and show the problem's sensitivity to a program's network utilization and data-sharing characteristics. Furthermore, we observe our approach's great potential in leveraging network energy efficacy (up to 21%) and system energy efficacy (up to 35%) in the experiments.

The rest of the paper is organized as follows. Section 2 illustrates power over/under provisioning in a network-metric based DVFS approach. Section 3 presents the overview and principles of our thread voting based DVFS technique. In Section 4 we detail the thread communication characterization with vote generation and propagation, as well as the vote counter organization and region DVFS controller for region V/F adjustment. In Section 5 we report experiments and results, and finally we conclude in Section 7.

2. MOTIVATION

We illustrate the power over/under provisioning problems observable in network-metric based DVFS mechanisms. To be specific, we exemplify the phenomenons using a router buffer load based DVFS technique. In this section, we also motivate the selection of region-based DVFS.

2.1 Target CMP architecture



(a) Hotspot traffic (b) Dispersed traffic

Figure 1: The target CMP architecture (NI: Network Interface and R: Router)

We first introduce the target CMP architecture we exploit through all experiments in the paper. Figure 1 shows a typical architecture for a 64-core CMP, where the NoC interconnects processor nodes (a CPU core with its private L1 cache), secondary on-chip shared L2 cache banks, and on-chip memory controllers together. Routers are organized in a popular mesh topology on which the XY dimensional routing algorithm is implemented to achieve simplicity and deadlock-free resource allocation. There are 8 memory controllers, which are connected to the central 4 nodes of the first and last row for architectural symmetry. In the figure we only illustrate 1 memory controller in the first and last row due to space limitation. For maintaining data consistency, a directory based MOESI cache coherence protocol is implemented. On the target CMP, once a core issues a memory request, first the local L1/L2 is checked for the presence of the requested data. If this fails, the request is forwarded to the corresponding remote L2/DRAM via the NoC.

2.2 Power over/under provisioning scenarios

Power over provisioning. We define *power over provisioning* as the circumstance in which the on-chip V/F level is highly increased but the application-level performance improvement is marginal. As illustrated in Figure 1a, assume that core 1, 2,..., 7, 8 periodically issues write requests to different data blocks within the same L2 cache bank A (such gather traffic pattern frequently occurs in multiplication of different partitions of a matrix). Assume further that the write requests are all exclusive and have no data sharer threads. In this case, the issuing cores can hide the write miss delay with write-buffers, and the return time of the request can be relaxed. Under this circumstance, although the traffic pattern forms a hot-spot in the highlighted router column, it may not always be appropriate to increase the routers' V/F level once their buffer load goes high, since the core can tolerate part of the network delay without loss of performance. However, in buffer load based DVFS, the frequency of routers in the highlighted column will be highly increased due to the hot-spot traffic but the performance improvement can be very limited.

Power under provisioning. We define *power under provisioning* as the circumstance in which the on-chip V/F is slightly decreased but the application-level performance degradation is very notable. As illustrated in Figure 1b, assume that core 1, 2,..., 7, 8 periodically issues write requests to L2 cache banks A, B,..., G, H, respectively. Assume further that each of the issuing thread has multiple data-sharer threads (such traffic frequently occurs in for example multiple threads competing to lock on shared variables). In order to maintain memory consistency, successive requests issued by the data-sharer threads to the same data address are prohibited from returning the newly written value until all duplicated cache copies have acknowledged the receipt of the invalidation or update messages. Under this circumstance, even though the traffic pattern is dispersed and each passing router may have light buffer load, it may not be proper to lower the router's V/F level, since the completion time of the ongoing write request affects both the issuing and data-sharer threads' progress. In router buffer load based DVFS, the frequency of the passing routers in Figure 1b will be improperly decreased due to low buffer load, which may sacrifice considerable application performance for limited power saving gains.

2.3 DVFS tuning granularity

The NoC DVFS control can be implemented in differ-

ent granularity, such as in the entire NoC chip [9, 23], in coarse-grained V/F island [3, 13], or in fine-grained per-router micro-architecture [20]. Although the chip-wide approach offers small inter-domain interfacing overhead and is often cheaper to implement, the recent trend towards CMPs executing multi-threaded workloads motivates the need for finer-grained DVFS mechanism [12, 20]. However, having each router in a separate voltage domain requires bridging of such domains on every single link, which will introduce 2-3 cycles of additional delay for each link [12]. Facing this dilemma, this paper tends to seek a hybrid solution to balance hardware complexity and DVFS tuning effectiveness: *region based DVFS tuning*, where several adjacent routers on the chip forms a DVFS region and share the same V/F level. Being different from the V/F island approach where V/F islands are formed dynamically over time, our approach adopts predefined static V/F regions for lower hardware complexity.

3. THREAD VOTING BASED DVFS MECHANISM

3.1 Design overview

As the motivational examples show, adjusting DVFS merely according to network metrics may lead to power over/under provisioning. What is equally important is to take data-sharing situations into consideration. With this in mind, we develop a DVFS mechanism exploiting both network utilization and data-sharing characteristics according to a program's runtime behavior. On the way to seek for a scalable solution, we develop a thread voting approach which, in contrast to thread-passive V/F turning, allows a thread to actively vote a V/F level to suit its own performance interest, and then a region-based DVFS controller makes V/F decision by the collected votes of involved threads.

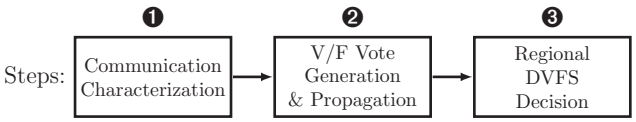


Figure 2: The three-step DVFS tuning mechanism

Technically, our voting based on-chip DVFS approach requires solving three essential problems: 1) How to capture a thread's communication characteristics reflecting both network utilization and data sharing? 2) How to generate and propagate the V/F vote into the NoC efficiently? and 3) How to make an effective DVFS decision according to the votes received? Figure 2 shows our technique which comprises three steps, with each step addressing one of the problems.

- Step 1 **Communication characterization** conducts a step window based thread communication demand characterization. Since per-thread communication characteristics are reflected by a thread's own message generation rate as well as its data sharing relations with other threads, we suggest a (γ, s) model where γ represents message generation rate and s level of data sharing.
- Step 2 **V/F vote generation and propagation** performs vote generating according to the profiled (γ, s) values. We treat votes from different threads with the same weight. In order to achieve a fair voting scheme, the total amount of votes a thread generates should be proportional to its network utilization level. Under this consideration, we use network packets to carry and propagate the votes into the network routers. The more packets a thread injects, the more votes the thread spreads and thus the more influence the thread has on the V/F election results.
- Step 3 **Region DVFS decision making**: each V/F region makes its own DVFS decision in favor of the majority based on the collection of received votes.

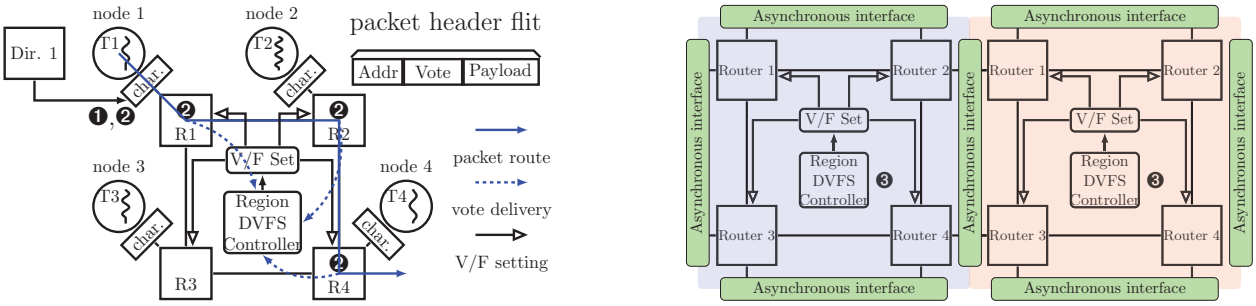
3.2 Per-thread communication need characterization using the (γ, s) model

The (γ, s) model. In the target CMP described in Figure 1, one key challenge lies in that threads running on core processors do not explicitly specify their on-chip communication demand. To adequately capture per-thread communication demand on the NoC, we define a (γ, s) model which can indicate not only a thread's own network utilization but also its data sharing relations with other threads. Specifically, γ is the message generation rate capturing the *average number of outstanding L1/L2 misses in miss status holding registers (MSHRs)* per epoch, and s is the *average number of data sharers per data request* per epoch as a thread's communication characteristic parameters. Because a core's L1/L2 misses must be encapsulated into packets and forwarded to the corresponding L2 or DRAM via the network, γ is indicative of the average network access intensity of the thread. Additionally, s reflects the criticality of a thread's data request with respect to other data-sharer threads. The more data sharers a thread has, the more critical the thread's data requests likely become, since the requests' finish time may influence the progress of the data-sharing threads.

Communication characterization. We profile per-thread (γ, s) values for each core at runtime. To obtain the γ value, we develop a communication *characterizer* within network interface (NI) to monitor the network requests from a thread, as shown in Figure 3a. Note that, since γ is defined to profile a thread's utilization of the NoC, only those cache behaviors that incur network messages are counted. Second, we use s to reflect the degree of a thread's data sharing status through characterizing the average number of data copies per epoch each thread has. Given a directory based cache coherence protocol, characterizing on s is done by checking the number of data sharers in the directory on outgoing network requests. Note that in order to allow continuously characterizing (γ, s) with a proper time granularity, we adopt a step-window based data sampling approach, where the (γ, s) values are computed on a per time interval basis. To ensure characterization continuity, adjacent windows may be overlapped.

3.3 Vote generation and delivery

As shown in Figure 3a, in Step 2 each thread votes for a preferred V/F level according to its profiled communication needs and then delivers its votes into the network for region V/F adjustment.



(a) Step 1 and 2 (b) Step 3

Figure 3: Overview of the thread voting based DVFS mechanism. (a) Steps 1 and 2 conduct per-thread communication demand characterization and vote generation/delivery. (b) Step 3 sketches the region based DVFS decision making.

Table 1: Mapping table from (γ, s) to a V/F vote

			γ		
			high	mid	low
read			Conf.A	Conf.B	Conf.C
write	s	high	Conf.A	Conf.A	Conf.B
		mid	Conf.A	Conf.B	Conf.B
		low	Conf.B	Conf.B	Conf.C

Vote generation. After characterizing the (γ, s) values, a thread votes for a preferred V/F level by mapping the obtained (γ, s) values to a proper DVFS configuration vote. For generality, we adopt a look-up table based approach for the mapping function, which is shown in Table 1. We divide the whole γ and s value span into, e.g., three intervals: `high`, `middle`, and `low`. We also pre-define, e.g. three V/F levels per region, which are denoted as `Configuration A`, `B`, and `C`, where `Conf.A` operates routers in a V/F region at maximum V/F, `Conf.B` 87.5% of maximum V/F, and `Conf.C` 75% of maximum V/F. As shown in the table, the combination of γ and s uniquely indexes a V/F configuration and thus a vote. For a fair voting scheme, the total amount of votes a thread generates should be proportional to its network utilization level. In general, one V/F vote can be generated on every nth read/write message from a thread. Note that, as shown in the table, since read accesses to shared data blocks do not affect data-sharing threads' progress, only γ is considered during the vote mapping on a read request. The intuition of the mapping rules in Table 1 is directed towards avoiding both power over and under provisioning. A proper V/F configuration is decided considering not only traffic metric γ but also data-sharing status s.

Light-weight vote delivery. The per-thread generated V/F votes have to be spread into the network in order to guide the region V/F adjustment. Although a *centralized* vote delivery approach is conceptually simple to design, it has limited scalability, i.e., the design overhead cannot be paid off when the system size enlarges to a certain degree. To amend this problem, we propose a *distributed* light-weight vote-delivery approach as follows. We treat the vote as payload, part of the head flit of a network packet. As shown in Figure 3a, specific bits in the header-flit are added for the vote. The number of the vote-bits depends on the number of supported V/F configurations. More added bits means finer V/F adjustment granularity, but also implies higher design complexity. With three candidate V/F configuration levels, we use 2 bits in the paper. We further illustrate the vote delivery progress in Figure 3a, where thread $T1$ spreads its votes to router R1, R2, R4 along the packet route.

3.4 Region DVFS decision making

In our mechanism, each region realizes the DVFS adjustment process in two phases: a *vote receiving* phase and a *DVFS decision making* phase. During the vote receiving phase, as packets are continuously routed through routers in a V/F region, the passing routers record the carried votes upon receiving the packet headers, and re-direct the votes to the Region DVFS Controller (RDC) for vote counting. When the vote receiving phase ends, the RDC begins the DVFS decision making phase, where the final DVFS adjustment decision is made in favor of the majority based on the number of votes received. The final DVFS decision is then input to the V/F setting logic for V/F level tuning.

In Step 3, the RDC consolidates the final DVFS decision from the collective statistics of the received votes. We maintain three counters, one for each V/F level, within each RDC to count the number of votes each candidate V/F level receives. In the DVFS decision making phase, the RDC compares the vote counters and sets the region V/F to the level denoted by the maximum counter, which represents the most elected V/F level by passing packets. Figure 3b shows Step 3, where two adjacent regions (each contains a 2×2 router mesh) are configured to two different V/F levels. Communication through different V/F regions are bridged by an asynchronous communication interface (ACI), which brings 2 cycles overhead. In an 8×8 mesh NoC with X-Y routing, the worst-case delay caused by the ACI is limited to at most 12 cycles since the longest path (network diameter) of the NoC consists of 7 different V/F regions.

In Section 4.1 and 4.2, we will introduce the design details of the thread communication demand characterization and the region-based DVFS adjustment, respectively.

4. DESIGN DETAILS

4.1 Characterization and Vote Generation

Our proposed DVFS mechanism relies on thread network access demand characterization for V/F vote generation and delivery. Figure 4 shows the micro-architecture of a modi-

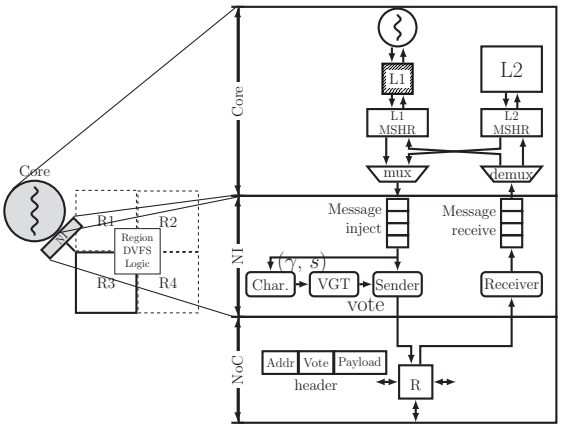


Figure 4: Modifications of the network interface, which conduct (1) communication demand characterization and (2) DVFS vote generation.

fied network interface where a *characterizer*, a *vote generation table* (VGT) and a *message sender* work cooperatively for this purpose. The characterizer monitors and profiles the network access demand of a thread periodically and profiles the (γ, s) values. The vote generation table generates V/F votes based on the profiled (γ, s) values and the message sender encapsulates the votes into outgoing packets.

Step-window based characterization. To allow continuously characterize the (γ, s) parameters with a proper time granularity, we adopt a step-window approach for the characterization. We design two windows: 1) the sampling window with length T_s, in which the parameters are sampled and characterized, and 2) the step window with length T_p, which controls the advancing step of the sampling window. To allow sampling continuity, $T_p \leq T_s$. The step window enables a flexible characterization scheme because one can adjust T_s and T_p to achieve quality-complexity trade-off. Specifically, let the ratio $\lambda = T_s/T_p$ defines the degree of overlapping among consecutive sampling windows. Indeed, a larger λ brings higher continuity of the consecutively characterized results but with higher computational complexity. To reduce computation complexity, we also set $T_s = 2^n$, where n is a natural.

Characterization of γ. To characterize the γ value, we have to examine how data messages are generated from a thread over time. In cache-based CMPs, network data message producing is always indicated by cache read/write misses. In state-of-the-art non-blocking/lock-up free caches, when an L1/L2 cache miss occurs, an access request to the L2/DRAM is created by firstly allocating an MSHR entry. The higher the number of outstanding misses in MSHRs for a thread, the more frequently the thread's data exchange occurs and the higher its data communication demand on NoC. Thus, the number of MSHR entries allocated by a thread directly reflects the traffic intensity of a thread from the NoC perspective.

For implementation efficiency, we keep the calculation of γ simple. At each sampling window T_s, two counters, namely c_1 and c_2, which are reset to 0 at the beginning of each sampling window, are maintained to record the total number of the MSHR entries for L1/L2 misses, respectively. When a new MSHR entry is allocated due to L1 or L2 miss, c_1 or c_2 increases by 1, respectively. At the end of each sampling window, γ_{L1} is computed for L1 cache miss rate by c_1/T_s, γ_{L2} for L2 cache miss rate by c_2/T_s, and γ for total cache miss rate by $\gamma = \gamma_{L1} + \gamma_{L2}$. To continuously characterize γ, the sampling window slides forward with a step length T_p after each characterization. Then the above procedure repeats through the system execution.

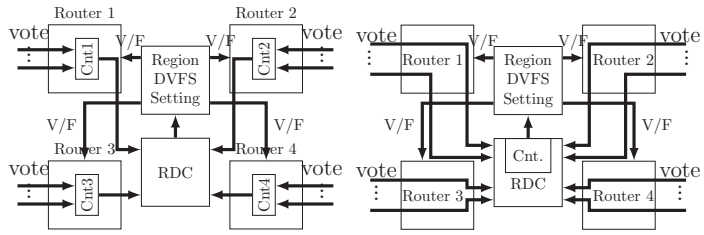
Characterization of s. The s profiling involves finding a thread's data sharing density, i.e., the average number of data sharers. This is done by performing a data-sharer number check on critical message m_c during a sampling window T_s, where the issuing time $t(m_c)$ of m_c satisfies $t(m_c) \in [0, T_s]$. A message is considered *critical* if it is a message accessing to a shared data block and its delay may affect the progress of other data-sharing threads. Such message can either be a cache miss or hit message, since in the write hit case the issuing thread must inform other data-sharing threads about the modification of a shared data block, and those data-sharing threads must wait for the completion of the write operation for memory consistency maintenance.

To profile s, we use a counter, namely c_s, to incrementally count the number of data sharers from each critical message. At the start of a sampling window, c_s is reset to 0. As time advances, the critical message check is performed at each message. If a critical message is detected, c_s is increased by s', which is the number of data-sharers associated with the critical message. Obtaining s' can be achieved by checking the coherence directory in a directory based cache coherence protocol, or by setting the s' value to the number of all destinations in a broadcast/multi-cast based protocol. In directory-based cache coherence protocol, the original request issuer is responsible for obtaining the number of sharers for critical messages. This requires that the request issuer fetches the number of data sharers from the directory controller that may exist in another node of the NoC. At the end of the sampling window, s is calculated by $s = \frac{c_s}{T_s}$, which denotes the average number of data sharers a thread has during the past sampling window.

V/F vote generation and delivery. As introduced in Section 3.3, after profiling the (γ, s) values, a thread generates votes for a preferred DVFS configuration by looking up the Vote Generation Table (VGT, as shown in Table 1 and Figure 4). New V/F votes are generated every nth critical cache read/write or coherence request transactions. For reply transactions, the reply packet follows the same vote of the corresponding request packet. This ensures that both the request and reply packets have consistent impact on the per-region DVFS decision making progress. To control the V/F adjustment sensitivity to (γ, s) values, we divide the γ and s value span into several levels, with each level denoting one candidate V/F configuration. Votes of a different V/F configuration start to generate only when γ or s shifts into a different level. The generated votes then get encapsulated into the packet header by the packet sender and spread into the network.

4.2 Region-based V/F Adjustment

We now introduce the design details of the region-based V/F adjustment in two parts. We first discuss vote counter organization and then present the architecture of the Region DVFS Controller (RDC).



(a) Distributed counter organization (b) Centralized counter organization

Figure 5: Distributed vs. centralized counter organization, where RDC refers to region DVFS controller.

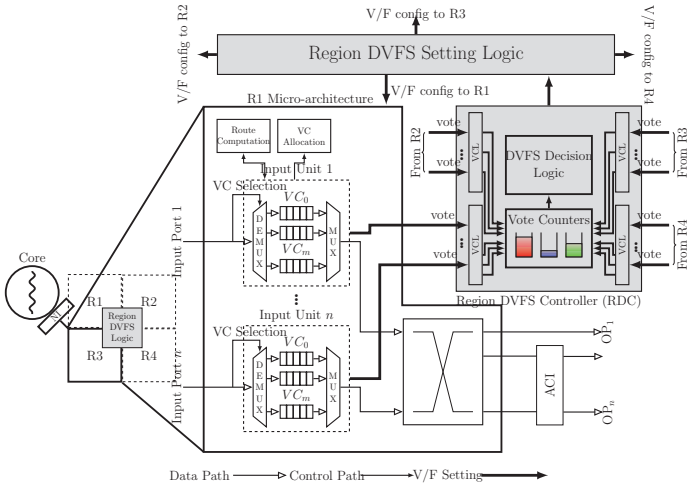


Figure 6: Router and region DVFS controller architectures

Vote counter organization and configuration. The vote counters within a V/F region record the number of votes each candidate V/F level receives. Figure 5 shows two vote counter organizations: the *distributed* organization and the *centralized* organization. In the distributed organization shown in Figure 5a, each router maintains its own vote counters. As packets are routed through the region, each router counts the received votes. At the beginning of the DVFS decision making phase, a 2-stage comparison is performed to find the maximum counter value. The first-stage comparison is performed locally in each router, then the local maximum values are output to the RDC for the second-stage comparison, based on which the final DVFS decision is made. Finally, the region DVFS setting logic adjusts the region V/F level based on the final DVFS decision. Although this approach keeps the RDC simple when the V/F region scales up, maintaining vote counters within each router adds however considerable hardware overhead.

Figure 5b shows the region-centralized counter organization, where centralized vote counters in the RDC count all the votes received by routers in the region. In the DVFS decision making phase, only a one-stage comparison is needed to find the regional maximum counter value. In our current implementation (each V/F region contains a 2×2 router mesh), we choose the centralized counter organization as it brings about most hardware implementation efficiency with fast counter comparison speed.

Region DVFS Controller (RDC) architecture. Figure 6 sketches the RDC architecture and its connection to one of the routers (Router 3) in region. As the figure shows, one RDC consists of four *vote counting logic* (VCL, one for each router), a set of *centralized vote counters* (one for each candidate V/F level), and a *DVFS decision making logic*, which work cooperatively to achieve the region based V/F adjustment. The router is a 2-stage speculative router [21] in which the first stage performs Route Computation (RC), Virtual Channel (VC) Allocation (VA), and Switch Allocation (SA) in parallel and the second stage is Switch Traversal (ST). In order to operate at different V/F levels, different V/F regions should be able to communicate asynchronously with each other. This is enabled by the asynchronous communication interface (ACI) on the boundary of the V/F region. We utilize the dual clock I/O buffer from [20] for the ACI implementation, which maintains independent read and write clock signals.

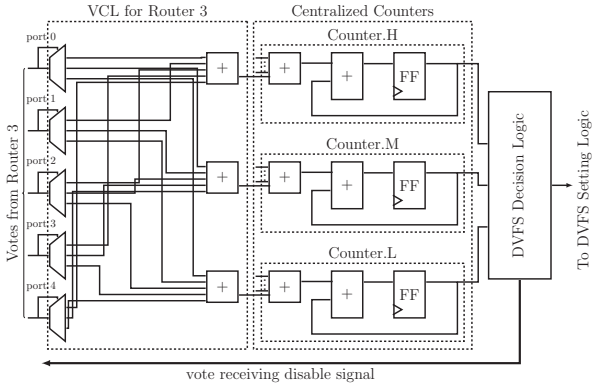


Figure 7: Schematic of the Region DVFS Controller (RDC)

Figure 7 draws the schematic of the RDC. For space restriction, we only show the VCL for Router 3 and its connection to the three centralized counters with `counter.H` counting high V/F level votes, `counter.M` middle V/F level votes, and `counter.L` low V/F level votes. The DVFS tuning procedure can be divided into two consecutive phases: a *vote receiving* phase and a *DVFS decision making* phase. Each router in the V/F region connects to the RDC through a VCL, which acts as the vote receiving interface. During the vote receiving phase, a router passes the received votes for a V/F level through VCL to the counter that counts votes for that V/F candidate. At the DVFS decision making phase, the DVFS decision making logic compares the vote counters and finds the maximum one. Then the region DVFS setting logic updates the region V/F to the level denoted by the maximum vote counter.

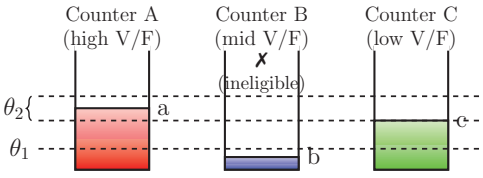


Figure 8: Comparison of vote counters. θ_1 denotes absolute threshold, and θ_2 denotes relative threshold.

To increase the comparison efficiency and reduce delay in the DVFS decision phase, we design two thresholds: an

Algorithm 1 V/F decision making

```
1: Input: centralized counters array cnt_array
2: Initialization: max_cnt ← 0, max_cnt_recorded ← False
3: Output: Updated region V/F level
4: if at vote receiving phase then
5:     if any cnt in cnt_array satisfies cnt ≥ ⌈(N×P×W+θ2)/2⌉ then
6:         max_cnt ← cnt
7:         max_cnt_recorded ← True
8:         Exit vote receiving progress
9: if at DVFS decision making phase then
10:    if max_cnt_recorded ≠ True then
11:        for each cnt in cnt_array do
12:            if cnt > θ1 and max_cnt < cnt − θ2 then
13:                max_cnt ← cnt
14: win_counter ← the corresponding counter with value max_cnt
15: Update region V/F to the V/F level represented by win_counter
```

absolute threshold θ_1 and a *relative* threshold θ_2, as illustrated in Figure 8. The absolute threshold θ_1 determines the eligibility of a counter. The intuition is straightforward: if a V/F level is seldom voted, it can be ignored in the final DVFS decision making progress. As shown in the figure, since very few packets vote the V/F level represented by counter B, we denote it as ineligible and only consider counters A and C during the DVFS decision making phase. The relative threshold θ_2 ensures that two counters are different only if their difference exceeds θ_2, which avoids comparing two close values and also enables flexible design trade-offs. For example, as shown in Figure 8, counter A (representing high V/F level) and C (representing low V/F level) satisfy $|a-c| < \theta_2$. The final DVFS policy can then be either biased towards high V/F level for better performance or towards low V/F level for better power saving, depending on the system running mode.

The pseudo code for the DVFS decision making logic is shown in Algorithm 1. Given that the DVFS tuning frame is W cycles, the maximum number of votes a region can receive during a time frame can then be determined by $N \times P \times W$, where N is the number of routers in the region and P is the number of input ports per router. During the vote receiving phase, if a counter excesses $\lceil \frac{N \times P \times W + \theta_2}{2} \rceil$, then no other V/F levels can receive more votes during the same DVFS tuning frame. The DVFS decision making logic records the V/F level as the majority selected one and disables the vote receiving signal (as shown in Figure 5b) to inform each router in the region to stop the vote receiving process. Otherwise, if no counter excesses $\lceil \frac{N \times P \times W + \theta_2}{2} \rceil$, the DVFS decision logic searches for the maximum counter value before making the DVFS decision, during which two threshold θ_1 (absolute threshold) and θ_2 (relative threshold) guides the searching progress as described in Section 3.

As shown in Figure 7, both the VCL and the centralized counters can be implemented with limited hardware resources. After synthesizing with Synopsis Design Compiler using 45 nm technology, one RDC consumes 697.7 standard NAND gates, which is about 3% area overhead averaged to each router. The power comsumption of RDC under 1.7 V/2.0 GHz condition is about 642.1 μW, with 634.2 μW dynamic power and 7.9 μW leakage power.

5. EXPERIMENTS AND RESULTS

5.1 Methodology

Experimental setup. Our DVFS mechanism is evaluated with well accepted simulators and benchmarks. We implement and integrate our design with the cycle-accurate full-system simulator GEM5 [5], in which the embedded network GARNET [1] is enhanced with our thread voting based DVFS mechanism. Further, we exploit McPAT [18] as the power measuring tool, in which the default system architecture is tailored to the configuration in GEM5, and the default technology is set to 45 nm. The details of the simulation platform setup are shown in Table 2. In the experiments, we divide the 64 routers into 16 DVFS regions, with each region consisting of 4 adjacent routers organized into a 2×2 mesh.

To count the additional power/area cost brought by the thread characterizer and the RDC, we synthesis our design with Synopsis Design Compiler (using 45 nm technology) under the three V/F levels in Table 2. The synthesis report shows that 64 characterizers and 16 RDCs consume additional power, which is 13.1 mW in 2.0 GHz, 12.5 mW in 1.75 GHz and 10.8 mW in 1.5 GHz, with 3% (11162.7 standard NAND gates) additional network area.

DVFS tuning window configuration. The length of the DVFS tuning window determines for how long a DVFS configuration is valid. In general, the shorter the tuning window is, the faster the DVFS adjustment becomes. However, the total system-level power saving may reduce due to frequent charge and discharge of voltage regulators. In this work, we use a pre-determined static tuning window, which is the same length as the thread behavior characterization window (as shown in Table 2) for lower complexity.

PARSEC configuration. To scale the PARSEC benchmarks well to 64 cores, we choose large input sets (`simlarge`) for all programs. Table 3 summarizes the problem size of the benchmarks. For data validity, we only report results obtained from the parallel execution phase (called as region-of-interest, ROI) in the experiments. Table 3 also shows the characteristics of PARSEC in terms of network utilization and data sharing degree among threads based on the data in [4]. As can be observed, four programs (*blackscholes*, *canneal*, *ferret* and *x264*) show consistent characteristics (both low or both high) in network utilization and data sharing, while the eight other programs exhibit inconsistent (one low one high or one high one low) characteristics in network utilization and data sharing.

Comparison study. For comparison purpose, we implement two baseline cases. The first case, which we denote *AHV* (Always High V/F), does not consider a DVFS mechanism and always operates the NoC at highest V/F level. The second case implements a router Buffer-Load based DVFS mechanism, denoted *BLD*, in which the NoC allows per-region V/F adaption by monitoring the averaged router buffer load in a region. Just like our Thread Voting based DVFS scheme (denoted *TVD*), *BLD* uses the same three (high, middle, low) V/F levels. When a gather traffic pattern (such as the case illustrated in Figure 1a) creates a hop spot in the network, the routers in the corresponding V/F region can adaptively switch to high V/F mode, increasing frequency to accelerate transmitting packets for performance assurance. When a disperse traffic pattern exists and the routers enjoy temporarily low buffer loads, the V/F region adaptively de-

Table 2: Simulation platform configuration

Item	Amount	Description
Processor	64 cores	Alpha based 2.0 GHz out-of-order processors. 32-entry instruction queue, 64-entry load queue, 64-entry store queue, 128-entry reorder buffer (ROB).
L1-Cache	64 banks	Private, 32 KB per-core, 4-way set associative, 128 B block size, 2-cycles latency, split I/D caches, 32 MSHRs.
L2-Cache	64 banks	Chip-wide shared, 1 MB per-bank, 16-way set associative, 128 B block size, 6-cycles latency, 32 MSHRs.
Memory	8 ranks	4 GB DRAM, 512 MB per-rank, up to 16 outstanding requests for each processor, 8 memory controllers.
NoC	64 nodes	8×8 mesh network. Each node consists of 1 router, 1 network interface (NI), 1 core, 1 private L1 cache, and 1 shared L2 cache. 4 routers form a DVFS region, in which a region DVFS controller implements the proposed DVFS policy and tunes the V/F level on a per time frame basis. X-Y dimensional routing. Router is 2-stage pipelined, 6 VCs per port, 4 flits per VC. 128-bit datapath. Directory based MOESI cache coherence protocol. One cache block consists of 1 packet, which consists of 8 flits. One coherence control message consists of 1 single-flit packet.
Thread char.	1/NI	65,536 (2^{16}) cycles of sampling window, 16,384 (2^{14}) cycles of step window. Overlapping ratio λ = 4.
DVFS ctrl.	1/region	16,384 (2^{14}) cycles of tuning window (the same size of a thread characterization step window). 3 supported V/F levels: 1.5V/1.5GHz, 1.6V/1.75GHz, 1.7V/2.0GHz.

Table 3: PARSEC benchmarks and characteristics [4]

Application	Problem size (all simlarge)	Characteristics	
		Network Util.	Data Sharing
blackscholes	65,536 options	low	low
bodytrack	4 frames, 4,000 particles	low	high
canneal	400,000 netlist elements	high	high
dedup	184MB data	low	high
facesim	80,598 particles, 372,126 tetrahedra	high	low
ferret	256 queries, 34,973 images	high	high
fluidanimate	5 frames, 300,000 particles	high	low
freqmine	990,000 click streams	low	high
streamcluster	16,384 points, 128 point dimension	high	low
swaptions	64 swaps, 20,000 simulations	high	low
vips	2,662×5,500 pixels	high	low
x264	128 frames, 640×360 pixels	high	high

creases the V/F level of the routers to low V/F mode for power saving. Refer to the configuration in Table 2, in our experiments the maximum buffer capacity of a virtual channel is 16 flits/cycle. If the region-wide average buffer load exceeds 12 flits/cycle (75% of maximum buffer load), *BLD* decides that the local congestion is high, and operates the routers in the region with high V/F level. If the region-wide average buffer load is below 4 flits/cycle (25% of maximum buffer load), *BLD* decides that the local routers are less loaded, and operates the routers in low V/F level. Otherwise, (with region-wide average buffer load goes between 4 to 12 flits/cycle), routers in the region operate in middle V/F level.

5.2 Experimental results

We report experimental results, first with detailed illustration of power over/under provisioning cases in PARSEC benchmarks and then the network and system energy efficacy improvements brought by our technique.

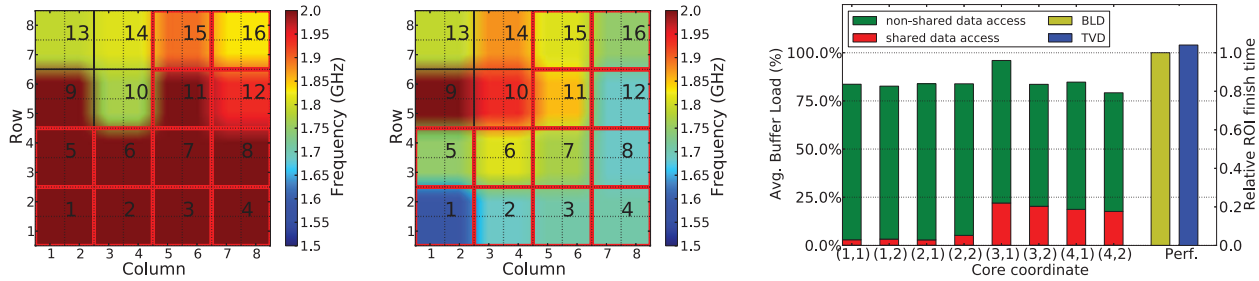
5.2.1 Power over/under provisioning in PARSEC

According to Table 3, we analyze possible power over/under provisioning to the benchmark programs by BLD. For *swaptions*, the BLD mechanism tends to create power over provisioning problem because the benchmark's network utilization level is high but the data sharing degree among different threads is low. The same goes for four other programs *facesim*, *fluidanimate*, *streamcluster*, and *vips*. For *bodytrack*, the BLD mechanism tends to create power under provisioning problem because the benchmark's network utilization level is low but the data sharing degree is high. The same goes for two other programs *dedup* and *freqmine*. In the following, we closely look into *swaptions* for power over-provisioning and *bodytrack* for power under-provisioning.

Power over-provisioning. Figure 9 demonstrates the occurrence of power over provisioning in *swaptions*, where the results are averaged over four consecutive V/F tuning windows (the same length as a sampling step window for thread characterization) in order to show stabilized V/F tuning effects. We number different V/F regions according to router coordinates and annotate the V/F regions where power over provisioning happens with red borders. As the figure shows, in the inspected DVFS windows, *power over provisioning has occurred in 12 out of the 16 V/F regions*. Figure 9a shows the effects of BLD, where the passing routers in the annotated V/F regions all boost to nearly maximum V/F level to accelerate packet transmission. Figure 9b shows the effects of our TVD, where routers in the annotated regions operate averagely at 85% of maximum frequency.

Figure 9c illustratively reveals 1) average router buffer load and the ratio between packets passing through routers in Region 1 (4 routers with coordinates (1, 1), (1, 2), (2, 1) and (2, 2)) and Region 2 (4 routers with coordinates (3, 1), (3, 2), (4, 1) and (4, 2)) to access *shared* data blocks and *non-share*d data blocks, 2) relative system-level ROI finish time in both *BLD* and *TVD* mechanisms. As the figure shows, all routers in region 1 and 2 suffer temporary high buffer load. However, most of the packets are accessing to non-shared data blocks. Thus they are non-critical and their delay can be tolerated by the issuing cores with several memory latency tolerance techniques such as read/write buffers [15, 14]. Figure 9c also shows that in terms of system-level performance, although the V/F levels in the 12 annotated regions are decreased by 15%, our *TVD* mechanism still achieves good system performance (only 4% slowdown in ROI finish time than *BLD*).

Power under-provisioning. Figure 10 shows the occurrence of power under provisioning in benchmark *bodytrack*, where the results are also obtained over four consecutive DVFS tuning windows. As the figure shows, *power under provisioning has happened in 14 out of the 16 V/F regions* in the inspected DVFS windows. Similar to Figure 9, we annotate the V/F regions where power under-provisioning occurs with blue borders. Figure 10a shows the effects of *BLD*, where the routers in the annotated regions averagely reduce to 85% of maximum frequency for power saving. Figure 10b

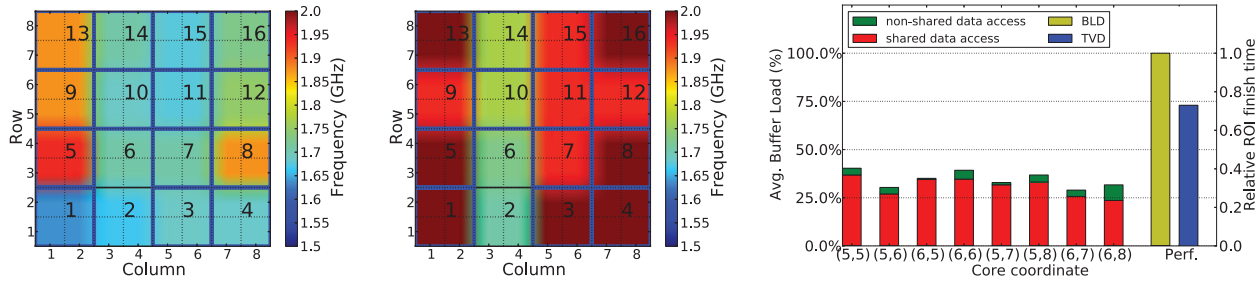


(a) Router buffer load based DVFS (BLD)

(b) Thread voting based DVFS (TVD)

(c) Avg. router buffer load and packets access ratio

Figure 9: Power over provisioning example in *swaptions*. (a) Network-metric based DVFS tuning results. (b) The proposed DVFS tuning results. (c) Average router buffer load and ratio between packets accessing to shared and non-shared data block.



(a) Router buffer load based DVFS (BLD)

(b) Thread voting based DVFS (TVD)

(c) Avg. router buffer load and packets access ratio

Figure 10: Power under provisioning example in *bodytrack*. (a) Network-metric based DVFS tuning results. (b) The proposed DVFS tuning results. (c) Average router buffer load and ratio between packets accessing to shared and non-shared data block.

shows the effects of our *TVD*, where routers in the annotated regions averagely boost to 95% of maximum frequency to accelerate packet transmission.

Figure 10c illustratively reveals the average router buffer load and ratio between packets accessing to *shared* data blocks and *non-shared* data blocks through V/F Regions 11 and 15. As the figure shows, all routers in Regions 11 and 15 enjoy temporary lower buffer load. However, most of the packets are accessing to shared data blocks, and their delay can further affect the progress of other data-sharer threads. Figure 10c also shows that in terms of system-level performance, although the V/F levels in the 14 annotated regions are increased by 10%, our *TVD* improves the system-level ROI finish time by 27% than the BLD approach.

5.2.2 *Network power and performance*

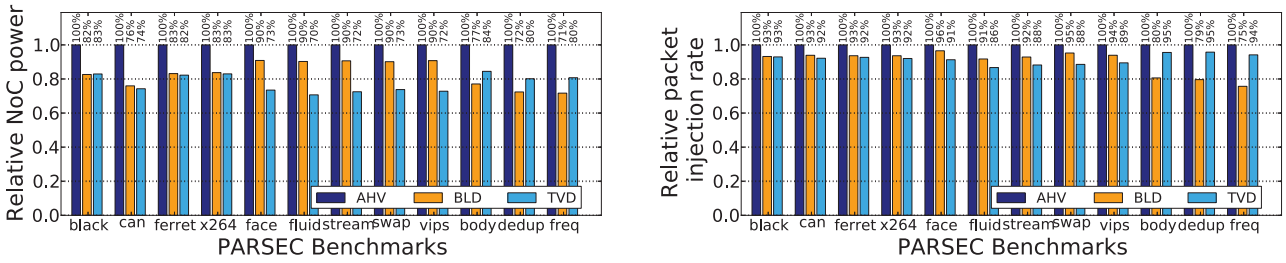
Before presenting the energy efficacy results, we compare the three schemes, i.e. *AHV*, *BLD* and *TVD*, in terms of network power consumption and network access rate.

Network power consumption. Figure 11a shows the network power consumption for the three comparison cases. In each benchmark, the results are normalized with respect to *AHV*, which consumes the most power due to constant high router V/F. In terms of network power consumption, *BLD* and *TVD* spend less power than *AHV* across all the PARSEC benchmarks, showing the power saving gains brought by DVFS mechanisms. However, the power saving of *TVD* against *BLD* depends on the network utilization and data-sharing characteristics of the benchmarks.

- In *blackscholes*, *canneal*, *ferret* and *x264*, both the network utilization and data sharing degree have the same characteristics (both are high or low). In this case, power over/under provisioning seldom occur, and *BLD* and *TVD* achieve similar power saving results.

- In *facesim*, *fluidanimate*, *streamcluster*, *swaptions* and *vips*, each benchmark's network utilization level is high but the data sharing degree among different threads is low. In this case, BLD tends to create power over provisioning problem since it increases on-chip V/F on high buffer load even if the on-going packets are accessing to non-shared data blocks. However, *TVD* alleviates this problem by providing lower V/F level and thus consumes averagely 20.0% less power.

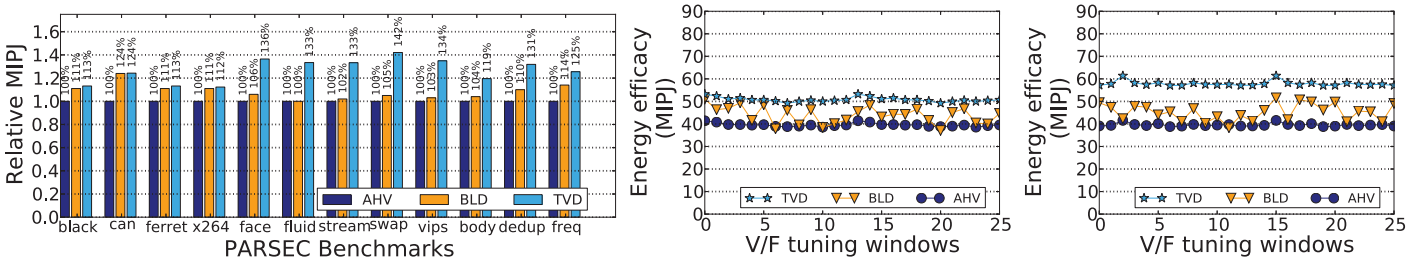
- In *bodytrack*, *dedup*, and *freqmine*, each benchmark's network utilization level is low but the data sharing degree among different threads is high. In this case, *BLD* tends to create power under provisioning problem since it lowers on-chip V/F on low buffer load even if the on-going packets are accessing to shared data blocks. As the figure shows, as an expense of alleviating this problem, *TVD* consumes averagely 10.9% more network power than *BLD*.



(a) Relative network power consumption results

(b) Relative network packet injection rate results

Figure 11: Network-level power consumption and packet injection rate results.



(a) Relative MIPJ results

(b) MIPJ comparison in *bodytrack*

(c) MIPJ comparison in *swaptions*

Figure 13: System energy efficacy results in terms of million instructions per Joule (MIPJ)

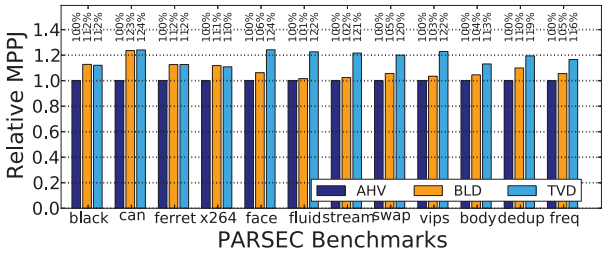


Figure 12: Network energy efficacy for all programs

Network packet injection rate. Figure 11b shows the network access rate results in the benchmarks. Since the NoC is self-throttling (a core cannot inject new requests into the network once it fills up all of its miss request buffers), the network packet injection rate directly correlates to the service speed of a core's miss requests and thus a core's performance. In each benchmark, the results are normalized with respect to *AHV*, which achieves the fastest network access rate across all benchmarks due to constant high V/F level. The figure also illustrates that in benchmarks where power over/under provisioning seldom occurs (*blackscholes*, *canneal*, *ferret* and *x264*), *BLD* and *TVD* achieve similar network access rate. In benchmarks where power over provisioning frequently occurs (*facesim*, *fluidanimate*, *streamcluster*, *swaptions* and *vips*), *BLD* achieves averagely 5.5% higher packet injection rate than *TVD*. Comparing with the additional (20.0% more) power *BLD* consumes, the improvement in packet injection rate is limited. In benchmarks where power under provisioning frequently occurs (*bodytrack*, *dedup*, and *freqmine*), *TVD* gains averagely 21.5% higher packet injection rate than *BLD* with 10.9% extra power.

5.2.3 *Network and system energy efficacy*

Network Energy Efficacy (NEE). Figure 12 depicts the relative network energy efficacy results measured in **MPPJ** (Million Packets Per Joule), which reflects the network's energy efficacy on serving packets from cores. As the figure shows, in *blackscholes*, *canneal*, *ferret* and *x264*, *TVD* behaves similarly to *BLD* since the power over/under provisioning problem is insignificant. However, *TVD* consistently achieves better NEE results than *AHV* and *BLD* in the remaining 8 out of 12 benchmarks, where the average improvement is 19.6% than *AHV* and 14.6% than *BLD*.

System Energy Efficacy (SEE). Figure 13a presents the relative system energy efficacy results in **MIPJ** (Million Instructions Per Joule), which measures the amount of energy expended by the CMP for executed instructions [2]. As can be seen, since DVFS increases the NoC power efficiency, both *BLD* and *TVD* increase the SEE against *AHV* across all benchmarks, where the average improvement in *BLD* against *AHV* is 8.4%, in *TVD* against *AHV* is 26.3%. Moreover, *TVD* avoids power over/under provisioning and constantly gives better SEE than *BLD* in all benchmarks. This reveals that both *AHV* and *BLD* consume more power than *TVD* for the same improvements in system-level performance.

Figure 13b and Figure 13c further illustrate SEE results in two benchmarks *bodytrack* and *swaptions* over different V/F tuning windows. As the figures show, the SEE curve of *TVD* lies above *AHV* and *BLD*. Since *BLD* tends to cause power under provisioning in *bodytrack* and over provisioning in *swaptions*, the SEE curve of *BLD* swings between that of *BLD* and *TVD* across different V/F tuning windows. This is not the case in both *AHV* and *TVD*. In *AHV*, the router V/F level remains constant. In *TVD*, power over/under provisioning are both effectively alleviated.

Table 4: Summary of energy-efficacy results

Application	Characteristics: Network Util.	Characteristics: Data Sharing	Leading probl. OP*	Leading probl. UP*	Improvement(%) MPPJ	Improvement(%) MIPJ
blackscholes	low	low	(non-opportun.)		0%	2%
canneal	high	high	(non-opportun.)		1%	0%
ferret	high	high	(non-opportun.)		0%	2%
x264	high	high	(non-opportun.)		0%	1%
facesim	high	low	✔		17%	28%
fluidanimate	high	low	✔		21%	33%
streamcluster	high	low	✔		19%	30%
swaptions	high	low	✔		14%	35%
vips	high	low	✔		18%	30%
bodytrack	low	high		✔	9%	14%
dedup	low	high		✔	8%	19%
freqmine	low	high		✔	10%	10%
Avg. of all benchmarks					9.7%	17.1%
Avg. of opportunistic benchmarks					14.6%	25.0%

*OP denotes power over provisioning and UP power under provisioning

5.2.4 *Summary of energy-efficacy results*

Table 4 illustrates the likelihood of power over/under provisioning in all benchmarks according to their network utilization and data-sharing characteristics. In 4 out of the 12 benchmarks where network utilization and data sharing degree among different threads have the same characteristics (both are low or high), there is little opportunity for TVD to outperform BLD because neither over- nor under- provisioning of power often occurs. Thus, we call these benchmarks as *non-opportunistic* benchmarks. However, the remaining 8 benchmarks are *opportunistic* where network utilization and data sharing degree have different characteristics (one low the other high or one high the other low). For these programs, TVD constantly delivers better energy efficacy in terms of MPPJ and MIPJ. As summarized in the table, the average improvements of TVD over BLD across all benchmarks is 9.7% in terms of MPPJ and 17.1% in terms of MIPJ. As expected, for those opportunistic benchmarks where power over or under provisioning problem frequently occurs, the average improvements of TVD against BLD increase to 14.6% in terms of MPPJ and 25.0% in terms of MIPJ, and the maximum improvement of TVD against BLD is 21% in terms of MPPJ and 35% in terms of MIPJ.

6. RELATED WORK

As shown in [12], on-chip DVFS can be implemented efficiently through on-chip voltage regulators in which the scaling delay can be reduced to tens of processor cycles, or on the order of the memory access latency. Such results justify DVFS for on-chip implementations. There are rich works in the literature which propose various DVFS techniques to increase energy efficacy in CMP systems. Often, these schemes rely on network metrics which indirectly reflect application/thread communication characteristics to guide the V/F adjustment in different granularity.

In [9], the entire NoC is considered as a single V/F domain to offer coarse-grain DVFS management. A centralized PI (proportional and integral) controller is used to dynamically adjust the V/F scale according to network throughput and average packet delay so as to minimize energy consumption under application performance constrains. Further in [23], treating the entire shared last level caches (LLC) and NoC as the CMP uncore, the authors combine PI controller with Artificial Neural Network (ANN) to achieve proactive DVFS, in which the PI controller bootstraps the ANN during the ANN's initial learning process. To capture the impact of the uncore upon overall system performance, [8] proposes an uncore DVFS approach using average network delay and memory access time as the V/F adjustment hints.

Work in [22] partitions a multi-core chip into multiple V/F islands. The authors then propose a rule-based DVFS control for each island according to both link utilization and router queue occupancy. Bogdan *et al.* [6] illustrates that computational workloads in multicores are highly complex and exhibit fractal characteristics. A characterization approach based on the dynamism of queue utilization and fractional differential equations has been proposed to guide on-chip DVFS in different V/F islands. In [20], Mishra *et al.* propose a fine-grained DVFS approach for NoC routers. They monitor router input queue occupancy, based on which the upstream router changes its V/F level. When a gather traffic pattern (like all-to-one traffic to a shared memory) creates a hot spot region in the network, the routers in the region can adaptively switch to a turbo mode, increasing frequency to accelerate transmitting packets for performance assurance. To guide per-core DVFS, memory request criticalty is considered in [19] for predicting DVFS's impact on system performance by identifying critical memory requests (*e.g.*, leading load requests) in a realistic DRAM model and predicting the network delay and the memory request finish time. Recently in [16], the authors propose to utilize highly predictable properties of cache-coherence communication to derive more specific and reliable NoC traffic predictions, based on which an improved V/F island based DVFS mechanism can be achieved.

When designing DVFS for NoCs, we consider application-level characteristics in message generation and data sharing which are directly linked to system performance. Furthermore, we leverage the role of cores or threads in actively influencing the DVFS decision through sending preferred V/F levels as "votes" to express their wishes to the region DVFS controllers, which make "democratic" decisions based on the majority vote for their network regions.

7. CONCLUSION

We have introduced a novel on-chip DVFS strategy that improves energy efficacy in CMP systems. To address the problems of power over/under provisioning in network metric based approaches, our DVFS technique adjusts NoC V/F levels according to per thread's direct performance indicatives in terms of message generation rate and data sharing metrics. The proposed DVFS adopts a light-weight voting based approach in which each thread votes a preferred V/F level according to its own performance indicatives. The V/F votes are then encapsulated into packet headers and propagated into the network. With a few routers composing a V/F sharing region, a region-based V/F controller collects all votes of passing packets and makes the final V/F decision by the majority vote in a "democratic" way.

By realizing and evaluating our technique with a 64-core CMP architecture simulated in GEM5 running all PARSEC benchmarks, we find that the workload-sensitive improvements in energy efficacy correlate consistently with the network utilization and data-sharing characteristics of the bench-

marks. There are no or marginal improvements in four programs (*blackscholes*, *canneal*, *ferret* and *x264*), because these programs have the same tendency in network utilization and data-sharing, being either both low or both high, and thus providing little opportunity to improve network-metric based DVFS mechanism. However for the other eight programs, our technique gives significant boost in energy efficacy, because these programs have inconsistent tendency in network utilization and data-sharing, being one low and the other high or vice versa, and thus offer salient opportunities for our technique to win over pure network-metric based DVFS approach. Indeed, compared to a router buffer occupancy based DVFS mechanism, our technique enhances the network energy efficacy in MPPJ by 9.7% on average and 21% in maximum as well as the system energy efficacy in MIPJ by 17.1% on average and 35% in maximum. We therefore believe that our technique gives a new insight (necessity of jointly considering message generation rate and data sharing property) and opens a viable approach (light-weight threading V/F voting with region-based DVFS) to designing highly efficient DVFS schemes for many-core networks.

8. REFERENCES

[1] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A Detailed On-chip Network Model Inside a Full-system Simulator," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009.

[2] M. Annavaram, "Energy per Instruction Trends in Intel Microprocessors," *Technology Intel Magazine*, 2006.

[3] A. Ansari, A. Mishra, J. Xu, and J. Torrellas, "Tangle: Route-oriented Dynamic Voltage Minimization for Variation-afflicted, Energy-efficient On-chip Networks," in *International Symposium on High-Performance Computer Architecture (HPCA)*, 2014.

[4] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Aug. 2008.

[5] 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 Computer Architecture News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[6] P. Bogdan, R. Marculescu, and S. Jain, "Dynamic Power Management for Multidomain System-on-Chip Platforms: An Optimal Control Approach," in *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2013.

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

[8] 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," *International Symposium on Network on Chip (NoCS)*, 2012.

[9] X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, and R. Ayoub, "Dynamic Voltage and Frequency Scaling for Shared Resources in Multicore Processor Designs," in *Design Automation Conference (DAC)*, Jun. 2013.

[10] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware Prioritization Mechanisms for On-chip Networks," in *International Symposium on Microarchitecture (MICRO)*, Jul. 2009.

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

[12] S. Eyerman and L. Eeckhout, "Fine-grained DVFS Using On-chip Regulators," *ACM Transation on Architecture and Code Optimization (TACO)*, 2011.

[13] S. Garg, D. Marculescu, R. Marculescu, and U. Ogras, "Technology-driven Limits on DVFS Controllability of Multiple Voltage-frequency Island Designs: A System-level Perspective," in *Design Automation Conference (DAC)*, 2009.

[14] J. Handy, *The Cache Memory Book*. Academic Press Professional, 1993.

[15] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach, Fifth Edition*. Morgan Kaufmann Publishers Inc., 2011.

[16] R. Hesse and N. E. Jerger, "Improving DVFS in NoCs with Coherence Prediction," in *International Symposium on Networks on Chip (NoCS)*, 2015.

[17] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and V. D. Wijngaart, "A 48-Core IA-32 Message-passing Processor with DVFS in 45nm CMOS," *Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, 2011.

[18] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *International Symposium on Microarchitecture (MICRO)*, 2009.

[19] R. Miftakhutdinov, E. Ebrahimi, and Y. N. Patt, "Predicting Performance Impact of DVFS for Realistic Memory Systems," in *International Symposium on Microarchitecture (MICRO)*, 2012.

[20] 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 *International Symposium on Microarchitecture (MICRO)*, Jul. 2009.

[21] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[22] A. Rahimi, M. E. Salehi, S. Mohammadi, and S. M. Fakhraie, "Low-energy GALS NoC with FIFO-Monitoring Dynamic Voltage Scaling," *Microelectronics Journal*, 2011.

[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)*, 2014.