LEAD: Learning-enabled Energy-Aware Dynamic Voltage/frequency scaling in NoCs

Mark Clark
Ohio University
Athens, Ohio
mc591611@ohio.edu

Avinash Kodi
Ohio University
Athens, Ohio
kodi@ohio.edu

Razvan Bunescu
Ohio University
Athens, Ohio
bunescu@ohio.edu

Ahmed Louri
George Washington University
Washington, D.C.
louri@email.gwu.edu

ABSTRACT

Network on Chips (NoCs) are the interconnect fabric of choice for multicore processors due to their superiority over traditional buses and crossbars in terms of scalability. While NoC's offer several advantages, they still suffer from high static and dynamic power consumption. Dynamic Voltage and Frequency Scaling (DVFS) is a popular technique that allows dynamic energy to be saved, but it can potentially lead to loss in throughput. In this paper, we propose LEAD - Learning-enabled Energy-Aware Dynamic voltage/frequency scaling for NoC architectures wherein we use machine learning techniques to enable energy-performance trade-offs at reduced overhead cost. LEAD enables a proactive energy management strategy that relies on an offline trained regression model and provides a wide variety of voltage/frequency pairs (modes). LEAD groups each router and the router's outgoing links locally into the same V/F domain, allowing energy management at a finer granularity without additional timing complications and overhead. Our simulation results using PARSEC and Splash-2 benchmarks on a 4 × 4 concentrated mesh architecture show an average dynamic energy savings of 17% with a minimal loss of 4% in throughput and no latency increase.

ACM Reference Format:
Mark Clark, Avinash Kodi, Razvan Bunescu, and Ahmed Louri. 2018. LEAD: Learning-enabled Energy-Aware Dynamic Voltage/frequency scaling in NoCs. In *DAC '18: DAC '18: The 55th Annual Design Automation Conference 2018, June 24–29, 2018, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3195970.3196068

1 INTRODUCTION

As technology scales further into the sub-nanometer region, an increasing number of transistors are packed onto chips. Single-chip processors already contain more than 7.2 billion transistors (Intel Broadwell-EP Xeon), and with this astronomical number of transistors comes several unique power challenges. Prior research has focused on reducing the excessive dynamic energy consumption resulting from storing and switching data within routers and links using Dynamic Voltage and Frequency Scaling (DVFS). As technology continues to scale down in size, static power consumption continues to grow, already accounting for a significant portion of the total power consumption of the chip. Power-gating [1] is a useful technique that seeks to save static power, however wakeup delay and break even time remain critical challenges.

The main goal when applying any DVFS strategy is the reduction of dynamic energy consumption at runtime [2], [3], [4], [5]. Because static power is not related to clock frequency, it is rarely considered, even though multi supply voltage designs assume that any increase or decrease in clock frequency is caused by a subsequent increase or decrease in supply voltage. This is because the supply voltage should be increased at times of high network traffic in order to increase throughput, and the supply voltage should be decreased at times of low network traffic in order to save dynamic energy. Various metrics have been used to measure network traffic such as the round-trip time (RTT) [5], Voltage Frequency Island (VFI) utilization [6], network slack [7], and buffer utilization [3]. The nominal supply voltage required for operation is hardware dependent, but the supply voltage and frequency always increase/decrease proportionally.

When dealing with any DVFS scheme, it is often the case that the logic behind when to increase or decrease the supply voltage becomes the most crucial part of the design (mode selection model). Recently some designs have begun to apply machine learning (ML) to their DVFS scheme in order to control the mode selection logic, thus determining when to switch modes proactively rather than older data-dependent reactive schemes [8], [9], [10], [11], [12], [13]. These approaches often apply online learning because it allows the algorithm to learn as data becomes available instead of learning from a static data set. However, online learning is expensive in terms of overhead cost as well as being inaccurate until after several iterations of learning.

In this paper we propose **L**earning-enabled **E**nergy-**A**ware **D**ynamic voltage/frequency scaling (LEAD), a collection of linear regression based DVFS techniques that are all trained offline. All LEAD models are proactive. They use only local router information when calculating the label and are trained offline in order to maximize the

DAC '18, June 24–29, 2018, San Francisco, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196068

reduction in overhead associated with traditional machine learning approaches. LEAD also scales the router and the router's outgoing links simultaneously in order to avoid inefficient use of network bandwidth or excess energy consumption. In order to achieve optimal energy-performance trade-offs, LEAD predicts different network metrics specific to the model. LEAD-τ predicts future buffer utilization, LEAD-∇ predicts the change in buffer utilization between the current and future epoch, and LEAD-G predicts change in $\frac{energy}{throughput^2}$ between the current and future epoch. Based on these predicted values, our proactive ML techniques are used to select the appropriate mode on a per router basis without the need for global coordination, thereby reducing the overhead and complexity. Machine learning uses linear regression algorithms that minimize the error of a model and make the most accurate prediction possible [14], [15], [16]. For a 4×4 concentrated mesh architecture, our simulation results show that LEAD-τ achieves an average of 17% savings in total dynamic energy with minimal loss of 2-4% in throughput for real applications.

2 RELATED WORK

There has been a significant amount of work in applying DVFS schemes to various on-chip components including the processor, caches, memory as well as the NoC. DVFS has also been applied at different levels of granularity ranging from very fine grained to very coarse grained. The trade-off between operating at a coarse-grain or a fine-grain comes in terms of system complexity and maximum amount of energy savings [7]. If the links operate at a much lower frequency than the router, packets can queue up and the network can saturate quickly. If the links operate at a much higher frequency than the router, then unnecessary dynamic energy will be consumed when little work is being done. Prior works have used various parameters to measure network traffic such as round-trip time (RTT) [5], VFI utilization [6], network slack [7], buffer utilization [3], or cache-coherence properties [17]. There has also been research focused on using different reactive DVFS mode selection models such as a threshold-based model, a proportional-integral (PI) based model, and a greedy model [6].

Recently, DVFS and machine learning have been combined such that the resulting regression based learning algorithm can automatically learn a DVFS scheme that maximizes dynamic energy savings within an allowable amount of performance degradation. While one approach has applied regression based learning to a heterogeneous embedded system [8], another approach has applied online reinforcement learning techniques [9]. One recent work has applied online learning to multi-tasking systems using only three entities: a controller, an expert, and the CPU [10]. It has also been shown that low-overhead reinforcement learning can be applied to multi-processor systems to achieve a required balance in temperature, performance, and power [13].

While prior work has applied reinforcement learning as well as regression models to the processor or shared computing resources, we specifically apply regression to NoCs to optimize dynamic energy and performance. Combined with offline learning, our proposed LEAD models greatly reduce overhead and implementation cost. Applying DVFS to the router and the router's outgoing links locally guarantees that our design has the necessary link bandwidth at times of high network traffic, while still saving dynamic energy when lower link bandwidth is sufficient.

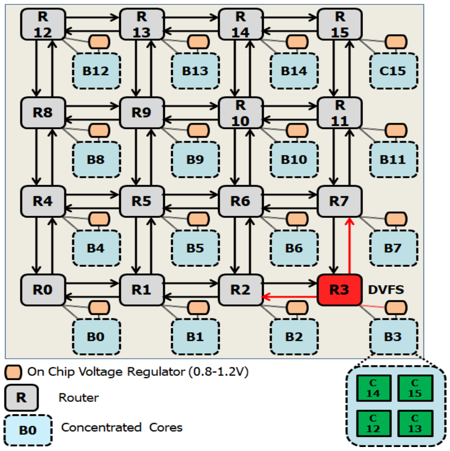

Figure 1: Topology: We apply LEAD to a concentrated mesh with 16 routers and 64 cores. We use on chip voltage regulators that can adjust the supply voltage between 0.8V and 1.2V, allowing us to apply DVFS to individual routers and their corresponding links.

3 LEAD TOPOLOGY

In this section, we will discuss the proposed LEAD topology, NoC microarchitecture, machine learning techniques and DVFS implementation.

LEAD Layout: LEAD is built on a concentrated mesh topology using on-chip voltage regulators that allow the selection of multiple voltage modes as shown in Figure 1. Our network consists of 16 routers, 64 cores, and 48 unidirectional links. We propose per router DVFS such that the router as well as the outgoing links are scaled simultaneously to operate at the same mode of operation. Each router consists of 8 input ports, 8 output ports, and 4 virtual channels per port. Each processor has an individual L1 cache and each router has an L2 cache shared among the four cores connected to each router. When a packet is first generated, the packet is stored in the input buffer. The output port is computed using XY dimension-order routing (DOR) in the route computation (RC) stage of the router pipeline. After a virtual channel is allocated, the head packet competes for the output channel in the switch allocation (SA) stage. After successfully competing and being awarded the channel, the packet is sent across the crossbar to the destination port in the switch traversal (ST) stage. The proposed router microarchitecture is shown in Figure 2(a).

Operating V/F Modes: Our models use five modes of operation with voltage and frequency levels similar to those proposed in previous work [17]. The supply voltage changes in 100 mV steps with proportional changes in clock frequency. The V/F pairs our models use include {0.8 V/1 GHz, 0.9 V/1.5 GHz, 1.0 V/1.8 GHz, 1.1 V/2 GHz and 1.2 V/2.25 GHz} which correspond to modes 1-5 as shown in Figure 2(b). We carefully chose five modes because using too many modes leads to increased voltage regulator overhead, whereas too few modes will not allow for dynamic energy reduction.

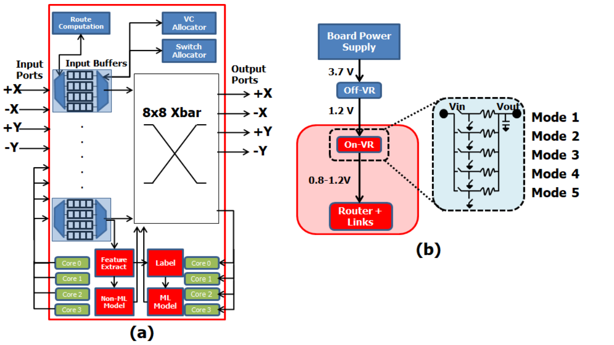

Figure 2: (a) Router Microarchitecture: The architecture as well as additional units required for reactive or proactive mode selection. Reactive model selection requires two units, Feature Extract and Non-ML Model. Proactive model selection requires three units, Feature Extract, Label, and ML Model. (b) VR Scheme: The on-chip voltage regulator setup that allows the selection of voltage levels in the range of 0.8V to 1.2V for every router and its' associated outgoing links.

Since power-gating has several challenges (deadlocks, breakeven time, loss in throughput), we have not considered implementing a power-gated version of LEAD, leaving this for future work.

3.1 DVFS Models

In this work we focus on measuring the impact of different mode selection models on dynamic energy savings and performance. We propose three machine learning based models; LEAD-τ, LEAD-∇, and LEAD-G. LEAD-G is based on an already proposed reactive model called Greedy. This Greedy model is presented in [6] as an adaptation of a Greedy search method presented in earlier work [18]. Greedy and LEAD-G are used strictly for comparative purposes.

Baseline: The baseline model always operates all routers in mode 5 (highest V/F) and does not apply DVFS. This model has the highest throughput and lowest latency, but has no dynamic energy savings.

LEAD-τ: LEAD-τ starts each router in the lowest mode. It chooses the routers' operation mode for the next epoch based on the predicted input buffer utilization of the router. If the router's buffers are predicted to be less than 5% full, then the lowest mode is chosen. If the buffers are predicted to be between 5% and 10% full, then the router will operate in mode 2. If the buffers are predicted to be between 10% and 20% full, then the third mode is chosen. If the buffers are predicted to be between 20% and 25% full, then the router will operate in the fourth mode. Finally, if the buffers are predicted to be greater than 25% full, then the router will operate in the highest mode. For larger epoch sizes the thresholds are reduced for more aggressive scaling. For example, at epoch size of 500, the thresholds are reduced to 1%, 2%, 4%, and 5% respectively because of how the maximum utilization is calculated as a worst case time variant sum. For simplicity sake we will show the thresholds as if they are all for epoch size of 100 cycles. This mode selection model assumes a voltage regulator scheme that allows the transition from any mode to any mode in one cycle without the need to transition into adjacent modes. This model emphasizes the importance of being able to select the optimal mode at any given epoch versus other designs which are constrained to only being allowed to transition into adjacent modes.

LEAD-∇: LEAD-∇ starts each router in the highest mode. Every epoch routers transition one mode up/down based on the predicted change in input buffer utilization between the previous and current epochs. Mode transitions only occur if this predicted change in buffer utilization falls within certain criteria. It must be carefully chosen so that small variations in network traffic do not govern mode selection, but also such that the router can adequately adapt to changes in network traffic patterns. The buffers must be predicted to increase by at least 6-10% of their maximum utilization in order to warrant a mode transition into a higher mode, and they must be predicted to decrease by at least 3-5% of their maximum utilization in order to warrant a mode transition into a lower mode. We ensure dynamic energy savings by requiring the predicted change in buffer utilization required to move down a voltage level be less than the change required to move up. This model is used to compare and contrast the trade-offs associated with being able to transition only into adjacent modes at every epoch, but we still assume that each transition takes one cycle. This model is more suited to gradual traffic changes where adjacent mode transitions are optimal.

LEAD-G: LEAD-G [6], [18] explores to find the mode that minimizes a predicted future $\frac{energy}{throughput^2}$. This model adds explorative logic and introduces both dynamic energy and throughput into the model in the hopes of better balancing the trade-off between the two. LEAD-G starts each router in the highest mode and in a downwards explorative direction. If the predicted change in $\frac{energy}{throughput^2}$ between the current and future epoch is less than or equal to 0, then the router will move one mode further in the current exploration direction (downward/upward). If the predicted change in $\frac{energy}{throughput^2}$ between the current and future epoch is greater than 0, then the router is put into a hold phase. The hold phase lasts 2 epochs, and during the holdphase the router can not change modes. After the hold phase expires, the exploration direction is flipped and the model begins to explore in the opposite direction until the predicted $\frac{energy}{throughput^2}$ is greater than 0 again. This model seeks to minimize the predicted $\frac{energy}{throughput^2}$ and assumes that routers may only transition into adjacent modes. The logic behind all three LEAD models is further explained in Figure 3.

DVFS Implementation: As shown in Figure 2(a), LEAD uses four components per router in order to perform reactive (non-ML) or proactive (ML) model selection. The first component is called Feature Extract. It gathers the router/link features and supplies it to the Label unit for proactive models. The next component is the Non-ML Model, which takes the label supplied from Feature Extract and selects the appropriate mode for the router and the router's outgoing links (Greedy model). For proactive model selection, we require the addition of two new units. The first is called the Label. This component takes the features supplied by Feature Extract and applies Ridge Regression in order to generate a corresponding label. This label is then supplied to the ML Model unit in order to select the appropriate mode for the router and the outgoing links.

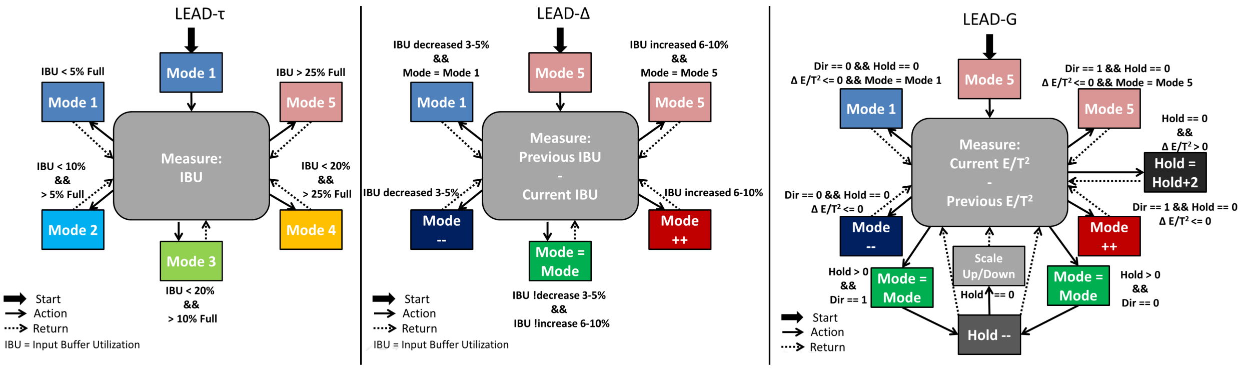

Figure 3: Mode Selection Models: LEAD-τ uses a predicted input buffer utilization to select the optimal mode per epoch. LEAD-∇ uses a predicted change in input buffer utilization to move in the direction of the optimal mode per epoch. LEAD-G uses a predicted change in $\frac{energy}{throughput^2}$ to move up/down adjacent modes based on exploration direction such that $\frac{energy}{throughput^2}$ is minimized.

3.2 Machine Learning Algorithm

We use machine learning to train different Ridge Regression algorithms corresponding to LEAD-τ, LEAD-∇, and LEAD-G. There are two arrays of values needed when using regression, the first being the feature set and the second being the weight vector. Each feature has a corresponding weight, a scalar factor representing that particular features impact on predicting the output. The Ridge Regression equation is shown below:

Ridge Regression:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$$

In the Ridge Regression equation listed above, we minimize the sum of squared errors between our predicted label $y(x_n, w)$ and the actual label t_n. The feature results x_n as well as the label t_n are used to train the system offline such that a corresponding weight vector w is created. The weight vector is normalized during training by adding L2 regularization. During tuning, different values of λ are tried for the equation $\frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$ until the best fitting solution for the training data is found. We used a total of 14 different trace files; 6 for training, 3 for validation, and 5 for testing.

Feature Set: The feature set is directly related to prediction accuracy and overhead cost. The feature set must be kept as small as possible because every new feature leads to an increase in arithmetic overhead. Our feature set is composed of 39 network parameters (buffer utilization, incoming/outgoing link utilization per direction, request/response packets, etc) as well as a label local to each of the 16 routers.

Label: A reactive version of each LEAD model is ran for each of the 6 training trace files. This is done so that corresponding feature results and labels can be extracted and supplied for training each LEAD model. While the same features are used to train all LEAD models, the label supplied for training is unique for each LEAD model. The label for LEAD-τ is the future input buffer utilization of the router for the next epoch. The label for LEAD-∇ is the difference between the routers' current and future buffer utilization. The label for LEAD-G is the difference between the routers' current and future $\frac{energy}{throughput^2}$. These labels are supplied along with the corresponding feature results in order to train the ML algorithm offline.

ML Overhead: A trained linear regression algorithm uses a series of additions and multiplies to calculate a label, thus the overhead cost can be simplified to the timing, power, and area cost required to execute a set number of additions and multiplies. The energy cost of a single 16 bit floating point add is estimated to be 0.4 pJ and the area cost is 1360 um^2 [19]. The energy cost of a multiply is estimated to be 1.1 pJ and the area cost is 1640 um^2 [19]. The total energy overhead cost is 58.1 pJ (considering two-stage multiplies followed by an addition), the total area overhead cost is 0.12 mm^2, and the total timing cost is 3-4 cycles. We use epoch sizes of 500 cycles and 1000 cycles, reducing overhead cost.

4 PERFORMANCE EVALUATION

Simulation Setup: In order to train our Ridge Regression model, we first begin by gathering features and labels from a cycle accurate simulator running a logically similar model with real traffic patterns. The first step in achieving this is to gather real network traffic trace files using Multi2sim. Multi2sim [20] is a full system simulator that uses benchmarks from PARSEC 2.1 [21] and SPLASH2 [22] in order to generate cycle accurate traces. These traces are used as input for our in-house network simulator such that feature results and labels can be extracted. We use the output of six trace files which include feature values and target labels in order to train our Ridge Regression algorithm in Matlab. The algorithm is then validated on three different trace files. This process is repeated for all three different LEAD models such that each has a uniquely trained Ridge Regression algorithm. It is important that we tune the lambda hyper-parameter as it controls the amount of L2 regularization used to combat over-fitting, resulting in reduced model complexity. After the models are trained and tuned, they are exported for testing back in our network simulator where they predict the various target labels. We use five untouched trace files as input during testing to measure the performance of our trained Ridge Regression models. These five traces are not used during testing or validation so that

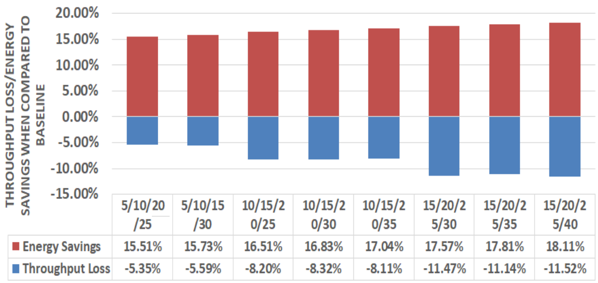

Figure 4: Throughput loss/dynamic energy savings across multiple Threshold selections for the lu trace with a window size of 500.

we can accurately measure the performance of each model. We used DSENT [23] to model routers and links so that the resulting dynamic power results could be converted to dynamic energy. This resulting dynamic energy is gathered for all 5 various modes at a 22 nm technology node assuming a 128-bit flit width. The total dynamic energy is calculated as the cost to send a specific amount of packets through the network. The energy per packet is calculated as the cost to traverse a router and an outgoing link which are operating at modes 1-5. This energy cost is listed starting from lowest mode to highest mode: {25 pJ/packet, 32 pJ/packet, 39 pJ/packet, 47 pJ/packet, 57 pJ/packet}.

Threshold: For the LEAD-τ model, we need to determine the ideal thresholds for switching between modes. In order to determine the ideal mode transition thresholds, we conducted an exhaustive search, a small set of which are shown in Figure 4 on one trace file. Each x-axis label has 4 values corresponding to the mode transition thresholds. For example, 5/10/20/25 implies a transition from mode 1 to mode 2 when buffer utilization exceeds 5%, from mode 2 to mode 3 when the buffer utilization exceeds 10% and so on. From the results, we observed that the combination that led to the best performance (least throughput loss and maximum energy savings) was 5/10/20/25 which yielded 15.51% energy savings while showing 5.35% throughput loss. We use 5/10/20/25 threshold for our LEAD-τ model.

Results: We compare the throughput/dynamic energy for all proactive LEAD models against a baseline model which does not apply DVFS and the reactive Greedy model as shown in Figure 5(a,b). Due to space constraints, we show the result for 500 and 1000 window cycles for LEAD-τ. We normalize the dynamic energy to the baseline for each application to make it easier to visualize the energy savings. At an epoch size of 500 and 1000 cycles, LEAD-τ reduces total dynamic energy consumption by 13-17% for a minimal loss in throughput of 2-4%. LEAD-∇ was able to significantly decrease dynamic energy consumption by 34-35% for a substantial throughput loss of 40-43%. LEAD-G performs the best in terms of dynamic energy saving which makes sense since it sought to move in the direction that minimized $\frac{energy}{throughput^2}$. LEAD-G was able to save a significant 42% dynamic energy for an almost even throughput trade-off of 42%. We see that LEAD-τ greatly improves total dynamic energy savings with a minimal loss in throughput. LEAD-G may not have performed very well in terms of throughput, but overall it did perform better than LEAD-∇ and was able to achieve an even trade-off between dynamic energy savings and loss in throughput. In a network with high amounts of contention, LEAD-τ would be the optimal mode selection model. However, if the network rarely became congested, then LEAD-G would be the best option. Figure 6 shows the amount of time spent in different modes across various applications. LEAD-τ shows that the ability to switch from any-to-any mode allows it to avoid mode 4 altogether. This ensures that LEAD-τ minimizes loss in throughput while maximizing energy savings. Both LEAD-∇ and LEAD-G show significant amounts of time spent in the lower modes 2 and 1 respectively. Therefore, both LEAD-∇ and LEAD-G tend to minimize energy at the cost of throughput. Table 1 shows the achieved mode selection accuracy for various applications for LEAD-τ model. Here, we measure the buffer utilization at the end of window and determine what mode should have been chosen. We compare this to the actual mode to evaluate if the prediction was accurate for that reconfiguration window. We achieve an average of 86% accuracy across all benchmarks with the canneal application showing almost 95% accuracy.

Table 1: LEAD-τ mode selection accuracy.

Models	Lu	Ls	Radix	Fluid	Canneal
LEAD-τ 500	88.3%	82.3%	62.7%	68%	95.9%
LEAD-τ 1000	83.2%	73.2%	56.6%	53.2%	95.9%

5 CONCLUSIONS

We have shown how smart proactive mode selection techniques such as LEAD-τ can lead to substantial dynamic energy savings, at times with minimal loss in performance. LEAD-∇ highlights how we can have an unequal trade-off between dynamic energy savings and performance, showcasing the opposite side of the spectrum; how a poorly tuned or otherwise underwhelming mode selection model can lead to sub-optimal dynamic energy savings/performance trade-off. LEAD-G shows a mode selection model that can get very good energy savings for an even throughput trade-off in saturated networks.

6 ACKNOWLEDGEMENT

This research was partially supported by NSF grants CCF-1054339 (CAREER), CCF-1420718, CCF-1318981, CCF-1513606, CCF-1703013, CCF-1547034, CCF-1547035, CCF-1540736, and CCF-1702980. We thank the anonymous reviewers for their excellent feedback.

REFERENCES

[1] L. Chen, D. Zhu, M. Pedram, and T. Pinkston, "Power punch: Towards non-blocking power-gating of noc routers," in *(HPCA-21)*, July 2015, pp. 378–389.

[2] A. Mishra K., R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. Das R., "A case for dynamic frequency tuning in on-chip networks," in *(MICRO)*, 2009, pp. 392–303.

[3] 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)*, October 2012, pp. 147–152.

[4] 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)*, May 2012, pp. 35–42.

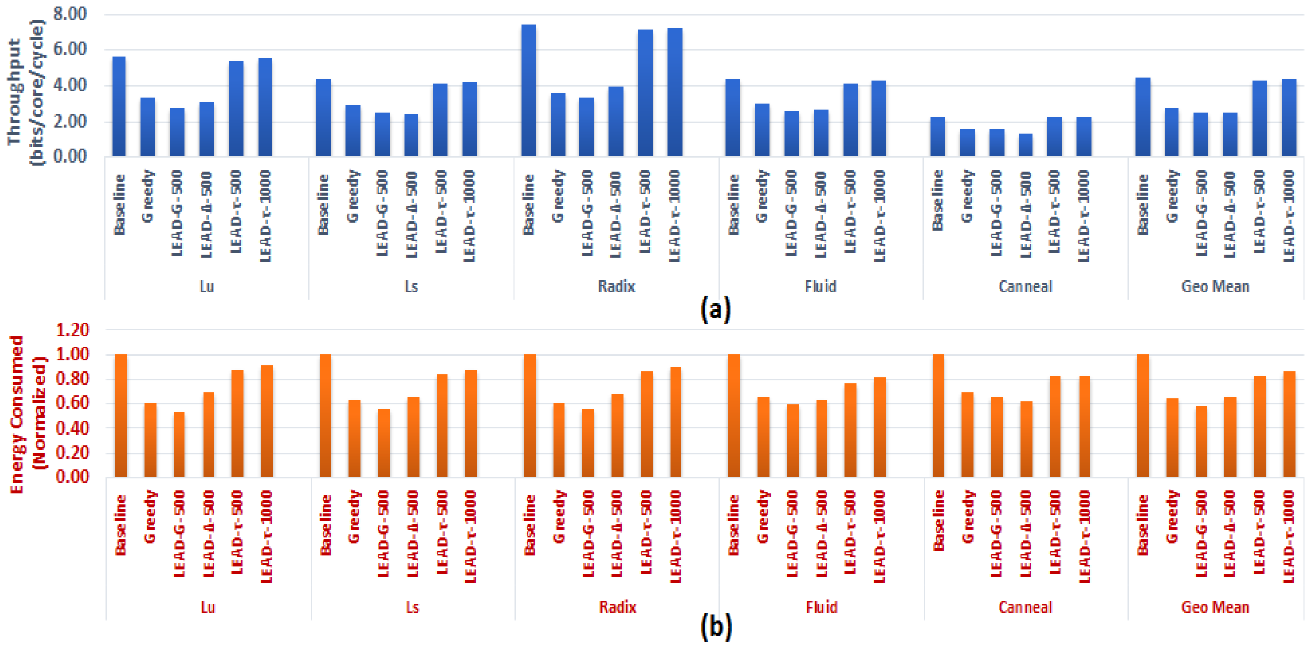

Figure 5: (a) shows the throughput for various LEAD designs compared against baseline and greedy. (b) shows the normalized dynamic energy. For LEAD-τ, we also show the result for 500 and 1000 window cycles.

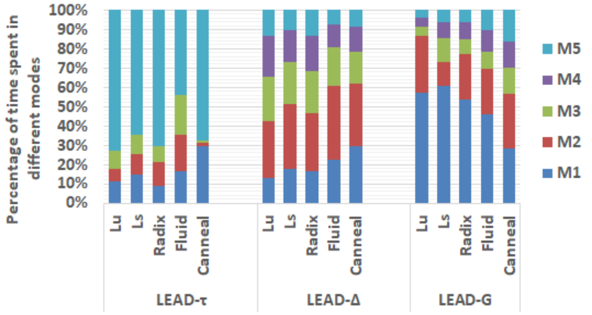

Figure 6: Time spent in different modes for LEAD.

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

[6] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *(ISLPED)*, August 2007.

[7] S. Eyerman and L. Eeckhout, "Fine-grained dvfs using on-chip regulators," in *ACM Transactions on Architecture and Code Optimization (TACO)*, April 2011.

[8] S. Yeng, R. Shafik A., G. Merrett V., E. Stott, J. Levine M., J. Davis, and B. Al-Hash M., "Adaptive energy minimization of embedded heterogeneous systems using regression-based learning," in *25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, September 2015.

[9] R. Jain, P. Panda R., and S. Subramoney, "Machine learned machines: Adaptive co-optimization of caches, cores, and on-chip network," in *(DATE)*, April 2016.

[10] G. Dhiman and T. Rosing S., "Dynamic voltage frequecy scaling for multi-tasking systems using online learning," in *(ISLPED)*, August 2007.

[11] H. Richard, "Machine learning based dvfs for energy efficient execution of multithreaded workloads," in *Dissertations and Theses Technical Reports-Computer Science*, November 2014.

[12] X. Chen, Z. Xu, H. Kim, P. Gratz V., J. Hu, M. Kishinevsky, U. Ogras, and R. Ayoub, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *(DAC)*, July 2013.

[13] H. Shen, J. Lu, and Q. Qiu, "Learning based dvfs for simultaneous temperature, performance and energy management," in *(ISQED)*, March 2012.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and S. R., "Dropout: A simple way to prevent neural networks from overfitting," in *Journal of Machine Learning Research 15*, June 2014, pp. 1929–1958.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[17] R. Hesse and N. Jerger, "Improving dvfs in nocs with coherence prediction," in *NOCS '15*, September 2015.

[18] G. Magklis, P. Chaparro, J. Gonzalez, and A. Gonzalez, "Independent front-end and back-end dynamic voltage scaling for a gals microarchitecture." in *(ISPLED)*, 2006.

[19] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 (ISSCC)*, February 2014, pp. 10–14.

[20] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2sim: A simulation framework for cpu-gpu computing," in *PACT '12*, 2012, pp. 335–344.

[21] C. Bienia and K. Li, " PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors ," in *Proc. of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[22] S. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *ISCA-22*, June 1995.

[23] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Networks on Chip (NoCS)*, 2012, pp. 201–210.