Machine Learning Enabled Power-Aware Network-on-Chip Design

Dominic DiTomaso, Ashif Sikder and Avinash Kodi
Department of Electrical Engineering and Computer Science
Ohio University, Athens, Ohio 45701
Email: dd292006@ohio.com, ms202708@ohio.edu, kodi@ohio.edu

Ahmed Louri
Department of Electrical and Computer Engineering
George Washington University, Washington DC 20052
Email: louri@gwu.edu

***Abstract*—Although Network-on-Chips (NoCs) are fast becoming pervasive as the interconnect fabric for multicore architectures and systems-on-chips, they still suffer from excessive static and dynamic power consumption. High dynamic power consumption results from switching and storing data within routers/links while excess static power is consumed when routers and links are not utilized for communication and yet have to be powered up. In this paper, we propose LESSON (Learning Enabled Sleepy Storage Links and Routers in NoCs) to reduce both static and dynamic power consumption by power-gating the links and routers at low network utilization and moving the data storage from within the routers to the links at high network utilization. As the network utilization increases from low-to-high, to accommodate more traffic, we design the same channels to flow traffic in either direction, thereby avoiding complex routing or look-ahead wake-up algorithms. Machine learning algorithms predict when to power-gate the channels and routers and when to increase the channel bandwidths such that power savings are maximized while performance penalty is minimized. Our results show that we can improve total network power consumption when compared to conventional NoC buffer designs by 85.6% and when compared with aggressive NoC buffer designs by 31.7%. Our predictor shows marginal performance penalties and by dynamically changing the direction of the links, we can improve packet latency by 14%.**

I. Introduction

With technology scaling down to the sub-nanometer regime, a majority of the Network-on-Chip (NoC) power consumption will be dominated by static power consumption [1], [2]. Static power is a result of leakage current and can vary between 30-90% of the total power consumption of the network depending on the network load [3], [4], [5]. As link utilization is typically low in real applications, varying between 20-30% [6], [7], [8], static power consumption normally consumes a majority of the network power. Therefore, static power is an important concern that must be addressed in current and future NoCs.

Researchers have proposed various techniques at all levels to improve power efficiency in Chip Multiprocessors (CMPs) while maintaining performance. Power-gating is a well-known technique for mitigating static power consumption [1], [3], [4], [9], [10], [11], [12], [13]. When NoC components such as buffers, routers, and channels are under-utilized, they can be shut down by disconnecting them from the voltage source. These components must be shut down long enough (break-even time) such that the energy savings is significant. However, if there is a sudden increase in utilization requiring more active components, then the wake-up time (typically 10-20 cycles [9], [1], [3], [10]) will impact performance. A way to deal with the wakeup issue is to utilize adaptive routing algorithms to route packets from source to destination through non-power-gated (possibly longer) paths. However, this would require prior knowledge of power-gated links or routers and moreover needs to create paths that are deadlock-free. Therefore, precisely determining the utilization of a component and the power-savings achieved by power-gating is critical for the success of the technique. Most power-gating techniques use basic network information to make an ad-hoc prediction about component utilization. Moreover, as buffers consume a substantial portion of the dynamic power consumption in NoCs, research has actively studied various techniques such as bufferless routers, moving the buffers from the router to the channel or using emerging technologies such as STT-RAMs [7], [14], [9], [15], [16]. However, most of the proposed techniques either target dynamic power or suffer from high performance penalties.

In this paper, we propose **LESSON** (*Learning Enabled Sleepy Storage Links and Routers in NoCs*) to reduce both static and dynamic power consumption by power-gating inter-router links at low network utilization and moving the storage from inside the routers to inter-router links as the network utilization increases. The key components of our proposed architecture are sleepy link storage (SLS) units that can save both static and dynamic power consumption while sustaining performance for various network load ranging from low to low-to-medium, to medium-to-high and high loads. At low network utilization, SLS are power-gated to save static power. At low-to-medium network utilization, as the network traffic increases, SLS can double as storage units to save dynamic power. At medium-to-high network utilization, SLS can switch the traffic direction to provide more bandwidth while saving both static and dynamic power consumption. At high network utilization, we optimize the performance by carefully storing data and reversing links to maximize power savings. Besides power savings at link and router storage levels, in this paper we also tackle the design of a power-efficient crossbar. To this end, we split the conventional uniform crossbar into several smaller crossbars which can be individually power-gated. Finally, to make accurate predictions, we propose to utilize machine learning algorithm such as a decision tree in order to make intelligent decisions. Decision trees have low overhead and can make accurate predictions by making comparisons on network information such as a history of link utilization and buffer utilization.

II. Related Work

As buffers in NoCs consume significant power and area, many designs have been proposed to reduce this buffering

978-3-9815370-8-6/17/$31.00

overhead such as bufferless routers [7], [14], elastic or channel buffering strategies [15], [16], and power-gating techniques [1], [3], [4], [9], [10], [11], [12], [13]. Elastic buffers (EB) have been proposed to reduce the buffering overhead at the router by using D Flip-Flops as FIFO buffers on the link in a flattened butterfly topology [15]. In QORE [16], multiple channel buffers are used between routers to reduce the dynamic power of buffering flits and reverse channels on demand to provide improved performance and fault tolerance. Our proposed SLS units not only reduce dynamic power and reverse channels, but also power-gate at low link utilization to save static power and improve performance by intelligently allocating channel bandwidth. Moreover, our work not only decides which link to power-gate but also considers power-gating individual crossbars within the router.

While power-gating has been implemented at the core/cache level of CMPs, more recently, power-gating has also been implemented at the network level [1], [3], [4], [9], [10], [11], [12], [13]. As the majority of the static power in NoCs is a result of buffers, many designs have focused on power-gating whole routers (e.g. buffers, crossbars, and control logic). For example, in order to keep the routers power-gated for longer than the break-even time, a bypass path is used so that intermittent flits that arrive can still traverse the router even when it is power gated in NoRD architecture [1]. In darkNoC [11], different NoC layers with different voltage-frequency levels are created such that individual layers can be selectively power-gated. Other power-gated designs utilize a fine-grained approach; targeting either individual buffers or other router components (e.g. crossbars, MUX, latches, etc.) [13], [4], [9]. Many NoC power-gating designs are limited by the wakeup latency and/or the inability to accurately predict the arrival of intermittent flits which makes the decision to shutdown or wakeup more complicated.

In networks with various voltage/frequency domains, finding the ideal operating point which maximizes energy efficiency with minimal performance overhead can be a complicated decision where machine learning algorithms could be utilized. Several design, however, have shown that learning techniques can be effective at optimizing this operating point [17], [18]. Machine learning algorithms can also be used to predict network congestion or hotspots in order to proactively employ preventive measures [19], [20]. With power-gating and channel allocation, prediction is critical in order to provide optimal power savings and high performance. In this paper, we use predictions from machine learning algorithms to make decisions about power-gating and channel allocation.

III. Architecture

A. Sleepy Link Storage Units

Figure 1 shows our proposed sleepy link storage unit (SLS). The SLS consists of an inverter, two additional transistors to enable storage, and four transmission gates to enable the change in direction on demand. The SLS units are also connected to a high threshold voltage transistor which is used for power-gating. The inverter is a standard gate consisting of an NMOS and a PMOS transistor which is used to propagate the signal down the link. Two additional NMOS and PMOS transistors, controlled by a 'Store' signal, are added to the inverter. When the store signal is activated, the two additional transistors are turned off, the signal is isolated on the link, and

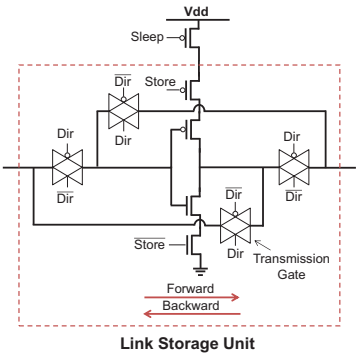

Fig. 1: Sleepy link storage unit with a high threshold voltage transistor used for power-gating and transmission gates used to change link direction.

the data can be stored on the link. Lastly, four transmission gates, each consisting of a pair of NMOS and PMOS transistors, are added to the link to be bi-directional and allow data transmission in either direction. When the 'Dir' signal is activated, the two appropriate transmission gates are activated and the other two are deactivated, allowing data to propagate in the opposite direction.

Similar to link repeaters, four SLS units are evenly spaced across each channel to ensure that the data reaches the downstream router within the link traversal cycle. A high threshold voltage transistor is then connected between Vdd and the SLS units to control the activation and deactivation of the link. Setting the threshold voltage high will result in a low leakage current transistor. When the 'Sleep' signal is activated, the high threshold voltage transistor cuts off the SLS units from the supply voltage which is the source of leakage current. Therefore, the SLS units can be power-gated at runtime by asserting and de-asserting the *Sleep* signal. These SLS units are critical in LESSON for reducing static power through power-gating and improving performance by changing the link direction.

B. Bandwidth Allocation and Power-gating Techniques

The proposed SLS units are the main components which enable both link reversal and power-gating in our proposed LESSON architecture. Figure 2 shows two routers with four pairs of SLS channels between the routers. Each SLS *channel* consists of four SLS *units* in series as shown in red squares. The storage space is similar to a conventional four flit buffer within the router. Each *link* consists of a pair of SLS channels, as shown within the dotted lines, to reduce head-of-line (HoL) blocking. To allow full connectivity between input ports and output ports, the SLS channels are connected to a 4x2 crossbar which is then input into our modular crossbars. The details of the crossbars are explained in Section III-C. Finally, there are four pairs of these SLS channels, or four links, between any two routers in our topology. At medium-high network utilization, these links can change direction when more bandwidth will be needed in order to improve performance. At low-medium network utilization, each link can be power-gated in order to reduce static power.

In our architecture, every N cycles, where N=100 in this paper, the links perform two main steps. First, the "direction"

controller for each link allocates a direction which will remain constant for at least the next N cycles. Second, the "power-gating" controller (shown as P-G controller) decides which link will be power-gated during the next N cycles. To allocate a direction to a link, the controllers must predict where the majority of the packets will be going. For example, when predicting traffic for the $\pm$x direction, the predicted label could be east (E), west (W), or both (B). The implementation of this predictor will be discussed in the next section. Using this predicted label, the direction controller allocates three out of the four links to the predicted direction or an equal number of links in each direction if the predicted label is B. The direction is set by sending a '*dir*' control bit to each of the four SLS links as shown in Figure 2. This *dir* control bit controls the transmission gates within each SLS which define the direction of the link. When links change direction, the existing flits inside each SLS will be flushed to escape buffers within the router. Each router port has a separate direction controller to reverse links for that port. In order to avoid disagreements, the controllers are placed in routers such that links are only reversed by a single controller. Lastly, the direction control signals are sent to the neighboring routers to notify these routers of the change in direction.

After the direction is decided for each link in the topology, the decision to power-gate the link follows. In parallel with the direction predictor, the link utilization is predicted. For example, when predicting the utilization of the +x links, the predicted label could be either low (L), medium (M), or high (H). Using this predicted label and the direction of each link, the decision to power-gate the links is made by the P-G controller. If the link utilization is predicted to be high then no links will be power-gated as the entire bandwidth will be needed to satisfy the applications' needs. If the predicted label is medium then up to one link is power-gated. Lastly, if the predicted label is low then up to two links can be power-gated. In order to maintain connectivity, when only one link is allocated to a direction then it will never be power-gated. As reversibility is an added benefit, we can continue to satisfy the bandwidth demands even when the remaining links are shut down. Once decisions have been made for all links, the "sleep" signals are sent to each link in order to cut off the supply voltage and shut down the link. Similar to the direction controller, there is a P-G controller at each router port and the neighboring routers are notified of the changes.

C. Modular Crossbar Design

Another source of power consumption is the crossbar. Typical NoC routers use a unified crossbar to switch flits from the input ports to the output ports. However, the power of unified crossbars scales poorly. By splitting the crossbar into several smaller crossbars, the power overhead can be reduced. The four SLS links from each direction are fed into 4 $\times$ 2 crossbars. These crossbars are designed by using 4 demultiplexers (DEMUXs) and 2 multiplexers (MUXs). After the 4 $\times$ 2 crossbars, escape buffers are next in the path and are used to store flits when the links reverse as explained in the previous section. Next in the path are the four modular crossbars. Each crossbar represents a routing quadrant: (+x, +y), (-x, -y), (-x, +y), and (+x, -y). For example, the (+x, +y) crossbar can allow flits to continue in the +x direction, can switch flits to the +y direction, or can send flits to the cores.

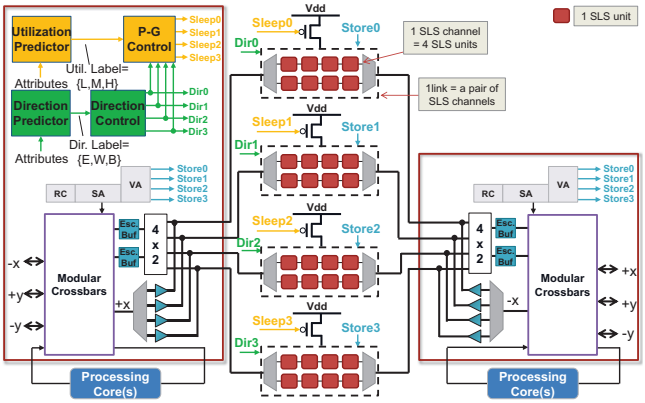

Fig. 2: Four pairs of SLS units between two routers with control for direction and power-gating.

Conflicts for output ports are handled by the switch allocator.

With these modular crossbars, we take advantage of the XY routing algorithm in that packets moving in one direction (e.g. +x) will never back track to the opposite direction (e.g. -x). Therefore, unused connections in a unified crossbar can be eliminated in our modular crossbars. However, we also add some redundant connections for each direction. For example, flits continuing their path in the +x direction have two crossbars to choose from, namely the (+x, +y) and (+x, -y) crossbars. Due to this redundancy, crossbars can be power-gated while still maintaining connectivity.

D. Router Power-Gating

To reduce the static power overhead of the routers in LESSON, we power-gate MUXs, DEMUXs, escape buffers, and whole crossbars. Each DEMUXs of the 4 $\times$ 2 crossbars can be power-gated at the same time the corresponding SLS channel is power-gated. The MUXs of the 4 $\times$ 2 crossbars and the escape buffers, on the other hand, are power-gated with the crossbar. The crossbars are power-gated independently from the SLS channels. However, the crossbars use outputs from the utilization predictor to make the decision to power-gate. Each crossbar is labeled X0-X3 and the initial state of each crossbar is ON. The algorithm checks the predicted utilization of the links which correspond to the directions in the crossbar's quadrant. For example, the (-x, +y) crossbar (X2) checks the utilization of links -x and +y. If the predicted utilization for both of the directions are less than high then the crossbar is eligible to be power-gated. To maintain connectivity in case of intermittent packets, not all crossbars can be power-gated. For example, if X2 is power-gated then X0 and X1 must remain in the ON state so that traffic moving in the -x or +y direction will not be blocked. Therefore, the decision to power-gate a crossbar is based on whether both directions have less than high utilization and the other crossbars with the same direction are on.

IV. Machine Learning for Power-Aware NoC

A. Decision Trees

Before the links can change direction or power-gate, two predictions must be made: 1) predict the link utilization to

Algorithm 1 Offline learning with the ID3 algorithm.

ID3(Training data D, Attribute A):
  if all samples in D have the same label:
    return a leaf node with that label
  if levels in the tree equals 3:
    return a leaf node with label chosen from a majority vote
  let $X \in A$ be the attribute with the largest information gain
  let R be a tree root labeled with attribute X
  let D_1, D_2, ..., D_k be the partition produced by splitting D on attribute X
  for each $D_i \in D_1$, D_2, ..., D_k:
    let R_i=ID3(D_i, A-$\{X\}$)
    add R_i as a new branch of R
  return R

determine when to power-gate, and 2) predict the traffic load for changing the direction of the link. The machine learning algorithm which is used for predictions is a decision tree. A decision tree is a flowchart-like model which consists of internal nodes, branches, and leaf nodes. The exact structure of a decision tree is determined in the learning, or training, process which can be done offline using the ID3 algorithm [21] as shown in Algorithm 1. A set of training data, D, and a list of attributes, A, are the inputs to the ID3 algorithm. At each node in the decision tree, the ID3 algorithm selects one attribute from the list A. The attribute selected as the root of the tree, attribute X, is statistically the best predictor as determined by the information gain metric. The terminating condition for the ID3 algorithm is that all samples in D have the same label. We add another terminating condition to this algorithm which limits the number of levels in the tree to three. This reduces the number of comparisons during testing, thereby saving latency and power.

After the decision tree is built, it can be implemented on the chip and used in online testing of new cases, or samples. Testing in decision trees involves testing attributes of the sample at each internal node and traversing the branches until a leaf, or decision is reached. The model we use for both predicting traffic direction and predicting link utilization is a decision tree due to the low overhead during the testing phase.

B. Predicting Link Utilization and Traffic Direction

In this section, we describe the details of implementing the decision trees for both predicting traffic direction and link utilization. To predict these two values, we engineer a list of attributes for the ID3 algorithm which consists of network information such as a history of link utilization, buffer utilization, and packet type (e.g. memory traffic such as request or response message which in shared memory multicores). Figure 3 shows some attributes used for router 0 (R0); however, these attributes can be generalized to any router. Shown in this figure are three of the sixteen routers in our proposed mesh topology; however, the proposed approach can be generalized to any other topology. Two links are shown connecting a pair of routers for illustrative purposes; however, in our architecture there are actually four reversible links between each router.

While designing the attribute list, we consider network information at R0 as well as the surrounding routers in the +x direction (router 1) and the +y direction (router 4). For example, the link difference attribute is calculated by

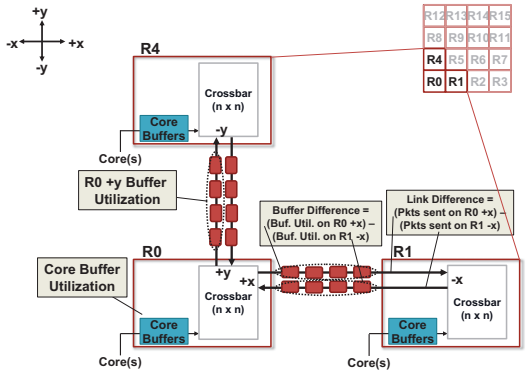

Fig. 3: Definition of attributes used to predict traffic direction and link utilization for router 0 (R0).

subtracting the number of packets sent on the -x link(s) of R1 from the number of packets sent on the +x link(s) of R0. The difference is then categorized into labels indicating where the majority of the traffic is going relative to R0: *In, Out, or Even*. Other attributes are calculated in similar ways and categorized into labels such as {*In, Out, Even*} or {Low, High}. These categorized labels are attribute values and are represented by branches in the decision tree. Since the ID3 algorithm selects the best attributes based on information gain, we engineer the full attribute list to include a wide range of network information. If any network information is irrelevant to the prediction, then it will be ignored by the ID3 algorithm.

V. Evaluation

In this section, we evaluate our proposed architecture in terms of network power and performance and compare it to other related architectures. For dynamic power, static power, and area estimates we use the DELPHI simulator [22] which combines synthesis of RTL hardware designs using Synopsys Design Compiler and models for energy and area from the DSENT tool [23]. We used a 22 nm technology node with a 1 GHz clock and a 0.8 V supply voltage. The dynamic power of a 128-bit SLS unit was found to be 0.216 mW and the static power was 0.374 mW. A conventional router buffer with the same storage space was found to have a dynamic power of 0.584 mW and a static power of 0.464 mW. A baseline 8x8 crossbar was found to have a dynamic power of 6.24 mW and a static power of 13.1 mW. Our modular crossbar design and four 4x2 crossbars was found to have a dynamic power of 12.3 mW and a static power of 16.6 mW. The timing of each component was found to be within our 1 ns clock period.

Our LESSON architecture is compared to a concentrated mesh network with conventional router buffers (CMesh), to a concentrated mesh with channel buffers (CMesh_CB), and to the QORE architecture [16]. Each design has 64 cores concentrated to 16 routers as this has been shown to minimize energy and latency while allowing a larger number of cores on a chip. Since our architecture is independent of topology and number of cores; results are similar for different network parameters. The packet size is four flits and each flit is 128 bits. For fair comparison, the bisectional bandwidth was kept equal across all designs. Traffic traces from real applications were collected from SIMICS with GEMS. Applications from the

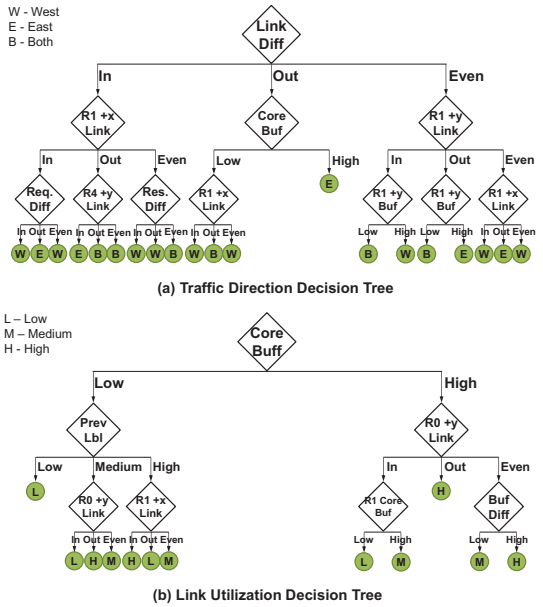

Fig. 4: Trained decision trees for (a) predicting traffic direction and (b) predicting link utilization on the +x link of router 0.

SPLASH-2, PARSEC 2.0, and SPEC CPU2006 benchmarks suites were used. We assume a 2 cycle delay to access the L1 cache, a 4 cycle delay for the L2 cache, and a 160 cycle delay to access main memory.

Decision Trees: Figure 4 shows the result of the trained decision trees for (a) predicting traffic direction and (b) predicting link utilization on the +x link of router 0. Each node is assigned an attribute as selected by the ID3 algorithm. The root of the traffic direction tree is the link difference from the last N cycles. The attribute values are *In, Out, and Even*. Every N cycles, the link difference of the last N cycles is observed and a branch is taken depending on the value of this attribute was. Attributes are tested and branches are taken until a predicted label is reached. Based on this predicted label, the decision to reverse links is made. The decision tree for predicting traffic direction was found to have an accuracy of 46.6% which is 13.2% higher than a predictor which randomly assigns labels. Similarly, the predicted outcome from the link utilization decision tree is obtained by traversing the tree until a label is reached. This label is then used to decide which links to power-gate. The decision tree for predicting link utilization was found to have an accuracy of 48.7% which is 13.8% higher than a random predictor.

The implementation of the decision tree in hardware consists of counters and tables to keep track of the attributes as well as comparators to test each attribute. The hardware cost is very low as three comparisons, at most, are required to reach a predicted label. Buffer utilization and link traversals can re-use existing hardware in the switch/virtual channel allocators and credit counters. The table that tracks attributes has 13 entries at most with each entry requiring less than 7 bits. Furthermore, the hardware is implemented in only half of the routers as a result of the disagreement avoidance discussed in Section III-B.

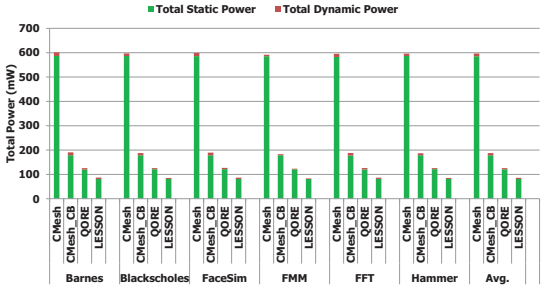

Fig. 5: Total network power of LESSON compared to other designs with a breakdown of dynamic and static power.

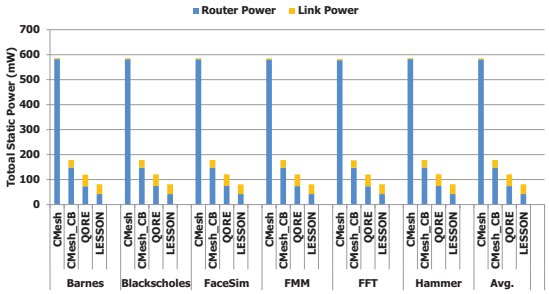

Fig. 6: Static network power of LESSON compared to other designs, broken down to router and link power.

Power: In this section, we evaluate the dynamic and static power of each design. Figure 5 shows the dynamic and static power across various real applications. With the low average network load of real traffic and the high leakage current of 22 nm transistors, static power dominates the total power as expected. On average, LESSON saves 31.7% total power over QORE, 54.3% total power over CMesh_CB, and 85.6% total power over CMesh. By moving the storage to the links, the SLS units in LESSON improve dynamic and static power consumption compared to an architecture with conventional router buffers. Additionally, by dynamically changing the direction of the links according to future demands, LESSON can execute applications faster than designs such as Cmesh and CMesh_CB. This results in components being on for fewer cycles in LESSON. Finally, by predicting link utilization 13.8% more accurately than a random prediction, LESSON can further lower power by intelligently power-gating during runtime of the application.

Figure 6 shows a closer look at static power by breaking the power down into router and link power. The static power of CMesh is highest due to the high static power of conventional router buffers. CMesh_CB requires more cycles to finish applications which results in higher overall static power. LESSON reduces both router static power and link static power due to the power-gating of links, escape buffers, and crossbars.

Performance: In this section, we evaluate the effects of LESSON on network performance. Figure 7 shows the average packet latency of the designs across various benchmark applications. The CMesh and CMesh_CB designs have the same topology but with different buffers. Therefore, the average packet latency is similar as expected. The reversibility in both LESSON and QORE can improve packet latency by approx-

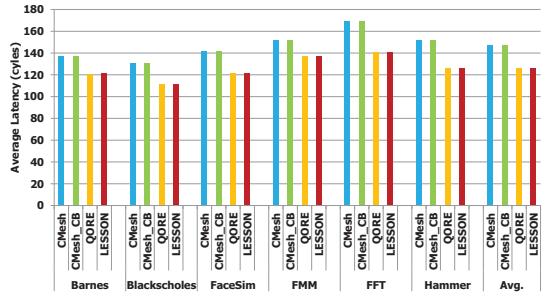

Fig. 7: Average packet latency of LESSON compared to other designs.

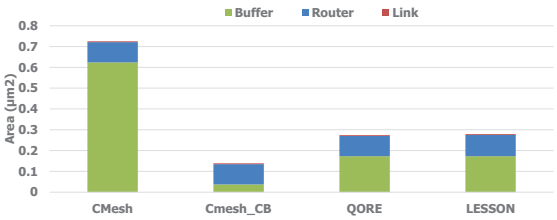

Fig. 8: Buffer, router, and link area breakdown of different designs.

imately 14% on average across all applications. Performing similar to QORE shows that power-gating links in LESSON has little performance overheads. Therefore, LESSON is accurately predicting link utilization and there are few times when packets require links that are power-gated.

Area: Figure 8 shows the area of the buffers, routers, and links in each design. Router and link area is similar across all designs due to the similarities in topology and low control overhead in LESSON. Due to the conventional buffers in CMesh, the total area of LESSON is approximately 62.3% less. The overhead of power-gating in LESSON only cause a 1.8% increase in area compared to QORE. The area of CMesh_CB is approximately half of LESSON due to the ability of LESSON to reverse links which improves performance.

VI. Conclusions

As transistor technology scales down, power consumption is becoming very critical in NoC designs, mainly due to buffers. In this paper, we propose LESSON (Learning Enabled Sleepy Storage Links in NoCs) to reduce both static and dynamic power consumption by power-gating the links and routers at low network utilization and moving the storage from routers to links. We design sleepy link storage (SLS) units which are capable of storing signals on the links, propagating signals in two directions with low power, and are able to shutdown on demand. Machine learning algorithms were utilized to predict traffic flow and link utilization. Based on these predictions, we can dynamically provide additional bandwidth when required in order to improve performance and we can accurately power-gate the links and buffers to improve power dissipation. Our results show that we can improve total network power between 31.7-85.6% and improve packet latency by up to 14% compared to other architectures.

Acknowledgment: This research was partially supported by NSF grants CCF-1054339 (CAREER), CCF-1420718, CCF-1318981, CCF-1513606, CCF-1547034, CCF-1547035, CCF-1565273, and CCF-1600820. We thank the anonymous reviewers for their excellent feedback.

References

[1] L. Chen and T. M. Pinkston, "NoRD: Node-router decoupling for effective power-gating of on-chip routers," in *MICRO-45*, 2012.

[2] R. Parikh, R. Das, and V. Bertacco, "Power-aware nocs through routing and topology reconfiguration," in *DAC-51*, June 2014, pp. 1–6.

[3] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," in *ISCA-40*, 2013.

[4] G. Kim, J. Kim, and S. Yoo, "Flexibuffer: Reducing leakage power in on-chip network routers," in *DAC-48*, June 2011.

[5] B. K. Daya, et. al., "Scorpio: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *ISCA-41*, June 2014.

[6] N. D. E. Jerger, L. S. Peh, and M. H. Lipasti, "Circuit-switched coherence," in *NoCs-2*, 2008.

[7] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ISCA-36*, June 2007.

[8] S. Borkar, "How to stop interconnects from hindering the future of computing!" in *Optical Interconnects Conference*, May 2013.

[9] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, "DimNoC: A dim silicon approach towards power-efficient on-chip network," in *DAC-52*, 2015.

[10] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *High Performance Computer Architecture (HPCA-19)*, 2013.

[11] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "darkNoC: Designing energy-efficient network-on-chip with multi-vt cells for dark silicon," in *DAC-51*, 2014.

[12] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of noc routers," in *High Performance Computer Architecture (HPCA-21)*, 2015.

[13] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Ultra fine-grained run-time power gating of on-chip routers for cmps," in *NOCS-4*, 2010.

[14] M. Hayenga, N. E. Jerger, and M. Lipasti, "Scarab: A single cycle adaptive routing and bufferless network," in *MICRO-42*, December 2009.

[15] G. Michelogiannakis, J. Balfour, and W. J. Dally, "Elastic-buffer flow control for on-chip networks," in *High-Performance Computer Architecture (HPCA-15)*, 2009, pp. 151–162.

[16] D. DiTomaso, A. Kodi, and A. Louri, "QORE: A fault tolerant network-on-chip architecture with power-efficient quad-function channel (QFC) buffers," in *High Performance Computer Architecture (HPCA-20)*, 2014.

[17] D.-C. Juan, S. Garg, J. Park, and D. Marculescu, "Learning the optimal operating point for many-core systems with extended range voltage/frequency scaling," in *Hardware/Software Codesign and System Synthesis*, 2013, pp. 8:1–8:10.

[18] J. Won, X. Chen, P. V. Gratz, J. Hu, and V. Soteriou, "Up by their bootstraps: Online learning in artificial neural networks for CMP uncore power management," in *High Performance Computer Architecture (HPCA-20)*, 2014.

[19] E. Kakoulli, V. Soteriou, and T. Theocharides, "Intelligent hotspot prediction for network-on-chip-based multicore systems," *IEEE TCAD*, vol. 31, no. 3, pp. 418–431, 2012.

[20] F. Farahnakian, et. al., "Q-learning based congestion-aware routing algorithm for on-chip network," in *NESEA-2*, Dec 2011, pp. 1–7.

[21] J. R. Quinlan, "Induction of decision trees," *MACH. LEARN*, vol. 1, pp. 81–106, 1986.

[22] M. K. Papamichael, et. al., "Delphi: a framework for rtl-based architecture design evaluation using dsent models," in *ISPASS*, 2015.

[23] C. Sun, et. al., "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *NoCS-6)*, May 2012.