Reinforcement Learning Enabled Routing for High-Performance Networks-on-Chip

Md Farhadur Reza*, and Tung Thanh Le†
Email: reza@ucmo.edu, tung.le@jdpa.com
*School of Computer Science and Mathematics, University of Central Missouri, Warrensburg, MO 64093, USA
†J.D. Power, Westlake Village, CA 91362, USA

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401790

***Abstract*—Network-on-chip (NoC) has been the standard fabric for multi-core architectures. With the increase in cores in the multi-core architectures, the probability of congestion increases because of longer path among sources and destinations in the NoC and because of the presence of multiple applications in a chip. Congestion hampers system performance because of the delay in packet delivery, which in turn results in reduced effective utilization of resources and reduced throughout. Higher congestion also results in higher energy consumption in NoC as packets spend more time in the network. Reactive detection and/or a single fixed routing algorithm are not effective to prevent congestion from happening for different traffic patterns in NoC. Therefore, we propose reinforcement learning based proactive routing technique that selects the best routing algorithm from multiple available routing algorithms using NoC utilization and congestion information to improve communication performance. Simulation results demonstrate latency performance improvement while providing robust NoC performance for different NoC states and traffic demands.**

I. Introduction

Due to the advancement of transistor technology, hundreds to thousands of cores have now been integrated on a single chip. For example, Cerebras has integrated 400K cores in a NoC based single chip [1], which can deliver more performance than a cluster of traditional machines. NVIDIA [2] has integrated 5.12K cores on a GPU chip. UC-Davis with IBM has integrated 1000 cores on a single chip [9]. However, communication is the major bottleneck to achieve high parallelism in systems with many compute nodes as communication costs much more time and energy than that of computation [6], [42], [51]. NoC has been the standard fabric for multiprocessor system-on-chip (MPSoC) and chip multiprocessor (CMP) because of its parallelism and scalability properties [35]. In a CMP or an MPSoC architecture, multiple nodes or tiles can be connected through a 2D-Mesh based NoC topology, where each node contains a local cache, a slice of the shared last-level cache, and a router for data/control transmission between the cores via varying bandwidth links.

Given a NoC architecture, routing becomes the most important design strategy for overall system performance. Routing provides a protocol for moving data through the NoC infrastructure and also determines the path of data transport. The selection of communication pathway would greatly affect the latency of packets transmitted from the source to the destination and, therefore, can have significant impact on the overall traffic flow in the network. Oblivious routing algorithms, such as dimension-ordered routing (DoR), do not take the network statistics into consideration for routing and so do a poor job of balancing the load across the NoC links. Most of the existing adaptive routing algorithms consider only local network state, leading to router output port selection based only on locally-available congestion estimates. Such short-sighted routing decisions tend to upset global load balance in many traffic patterns. Adaptive routing algorithms that consider global NoC state incur hardware overhead. However, dynamic routing is desirable because of its substantial improvement in communication bandwidth and intelligent adaptation to congested traffic. Furthermore, due to dynamic nature of the applications and introduction of new application with unknown demands, a single routing algorithm may not be enough for routing packets for high-performance. For example, traffic demands can change significantly at different times of a day.

An intelligent routing mechanism is required to minimize latency to balanced the loads and to avoid congested paths. Machine learning (ML) enabled agent can predict good routing paths and routing algorithms depending on the past data and training of the model. Reinforcement learning (RL) is suitable for dynamic environment as it can take into account the runtime changes in the environment. The most important characteristic of RL is that it can take best action by considering the delayed consequences (long-term impact) or utility of the actions. This property makes it a perfect candidate for dynamic environment of NoC. In this paper, we introduce a novel methodology to enable network-level RL enabled routing in a NoC. The major contributions of this work are outlined below:

- *Dynamic NoC Routing using RL:* Routing algorithm is dynamically decided for whole network (all the routers) depending on the learning of NoC traffic and congestion using RL at runtime to improve the NoC performance.
- *Evaluation on Real System Simulator:* Our proposed approach is evaluated using 64-core NoC architectures on a real system simulator. Synthetic traffic is used for evaluating the proposed approach compared to existing solutions. Simulation results show that the proposed RL enabled routing approach improves the performance compared to traditional non-ML based solutions while the proposed solution is robust to handle any kinds of traffic demands in NoC.

The paper is organized as follows. We discuss the related work on NoC routing/configuration and ML solutions in Section II. The details of RL model and components for congestion prediction and proactive NoC routing is presented in Section III. Simulation results are presented in Section IV.

II. Background and Prior Work

NoC buffers and links status are two of the most popular ways to indicate the existence of network congestion. In NoC, the routing of packets or parts of packets, namely flow control digits (flits), determines the inter-node communication performance, including congestion. Routing strategies can be categorized into deterministic and adaptive schemes. In a deterministic routing strategy, source and destination determine the traversal path. Popular deterministic or congestion-oblivious routing scheme for NoC is XY-routing, which is referred to as DoR [13]. [30] examined buffer availability at adjacent routers as a congestion metric. [43], [44] use the queue length at each output port as its local congestion indicator during the routing phase. [23] integrates both non-local and local congestion information for routing decisions so that each router has a better picture of network hotspots. At each network hop, the router aggregates its local congestion estimate with that of neighboring nodes. This approach uses a monitoring network to propagate congestion information among adjacent routers. There are several other works on adaptive routing in on-chip networks using traditional heuristics [4], [22]. Duato's theory [17], [18] is widely used in the design of fully adaptive routing algorithms. Unlike previous works, we propose to use different routing algorithms at different times depending on traffic demands and NoC states. We propose ML to help with the routing algorithm decisions depending on congestion and performance information in NoC.

ML techniques, such as regressions and neural networks (NNs), are being explored for optimization in on-chip systems. [31] proposes to use automated data-driven framework to quickly configure and design manycore systems for a wide-range of application and operating scenarios. The authors proposed to use ML for both design-time and run-time decisions to create fully-adaptive systems that are holistically optimized across the entire design stack. [40] proposes RL technique to proactively configure the NoC link-bandwidths in heterogeneous architectures for energy-efficiency. NoC link-bandwidths are configured dynamically based on the learning of the system (e.g., link utilization, buffer utilization). [39] proposes RL to configure the NoC voltage/frequency (V/F) levels depending on demands of the tasks and neural networks is used to approximate Q-values for different V/F actions. [50] presented a deep RL approach for efficient NoC arbitration. The proposed self-learning decision making mechanism reduces packet latency, which results in improved NoC throughput. [32] proposed an imitation learning (IL) based methodology for dynamic V/F island (cluster of nodes/links) control in manycore systems. [28] presented an NN-based intelligent hotspot prediction mechanism that was used with a congestion-control mechanism to handle hotspot formations efficiently. Here the NN uses online statistical data to dynamically monitor the NoC interconnect fabric, but reactively predicts the location of hotspot(s). [41] proposed runtime predictive configuration of node voltage-levels and link widths of NoC using NNs for energy-savings while addressing both power and temperature constraints of manycore NoC. [33] introduced a multi-layer NN based predictive routing algorithm which uses network state and congestion information to estimate routing costs and perform low-latency routing in NoC. A few other related works proposed for the design and optimization of NoC and/or multi/many-core systems using machine learning are: [5], [8], [11], [12], [14]–[16], [21], [24]–[27], [34], [36]–[38], [46]–[49], [52].

However, there has not been significant number of works that focus on ML for adaptive NoC routing for congestion avoidance. [29] proposed RL-based routing solutions to improve NoC throughput. Their paper demonstrated the promising ability of using RL to optimize the runtime NoC performance for different traffic patterns. [20] proposes RL based congestion-aware adaptive routing algorithm in NoC. The proposed dual Q-routing method provides the routing policy to alleviate congestion in the network by estimating latency values between each pair of source and destination nodes of the network. [19] also proposed a similar solution for routing packets on less congested paths using RL. Our work focuses to dynamically route packets using different routing algorithms to reduce congestion to improve the NoC performance. The main advantage of our work is that it can meet the real time requirements of the systems that need to satisfy heterogeneous demands of the application(s).

III. Reinforcement Learning Enabled Routing

Multiple paths exist between a source and a destination in NoC. So, NoCs must implement a routing algorithm to route packets to their destinations. A link in the NoC can carry different amount of traffic. It can happen that some links are over-utilized and some links are under-utilized. Over-utilization leads to network congestion, and then network congestion leads to packet blocking problem. DoR routing algorithms, which do not consider congestion, can itself introduce load-imbalance in the network. Because of the load-imbalance, congestion and latency of the packets increase. Higher latency leads to less throughput, less effective utilization of the NoC resources and higher energy consumption. To achieve better performance through adaptive routing, a router ideally needs global knowledge of the current network congestion status. However, due to hardware overhead and complexity, dynamically obtaining a global and instantaneous view of the network is often impractical. To address the limitations of a single DoR or adaptive routing algorithms, we propose to use different routing algorithms for different traffic patterns using RL, where an agent is trained with global NoC knowledge and can take global optimal action to improve NoC performance, as shown in Fig. 1. An RL agent is used to observe the NoC states and takes routing decisions globally for all the NoC routers depending on traffic demands. RL technique is greatly suitable for quick decision in dynamic environment because of its autonomic properties. RL can change its model within a quick time with the dynamic changes in the resources of the system and/or changes in the demands of the applications, where a supervised ML technique needs complete retraining to adapt to changes. Our goal is to develop (train) an RL agent, which can choose optimal routing algorithms at different traffic demands for high-performance NoC.

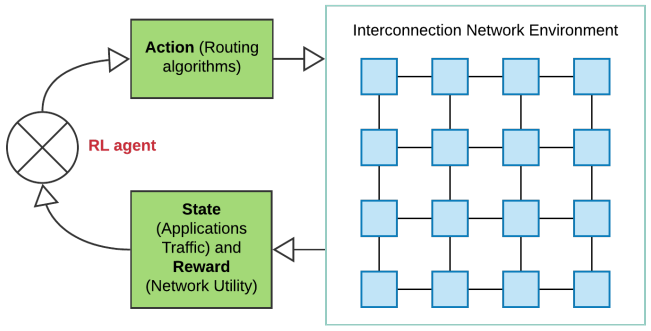
Action (Routing algorithms)
Interconnection Network Environment
RL agent
State (Applications Traffic) and Reward (Network Utility)

Fig. 1: Proposed RL solutions for NoC architectures.

In this work, we are proposing to use predictive routing algorithms. Predictive routing exploits the path diversity of the NoC topology to select a less congested path for every source-destination pair. Predictive techniques are efficient at routing traffic around congested hotspots, balancing load across the network, thereby providing better performance. The main objective of the proposed predictive RL-based routing algorithm is to minimize congestion by sending packets through paths with least latencies. At different injection rates of packets, traffic demand on NoC links changes and NoC links face heterogeneous delay and contention. To route the packets quickly, different routing algorithms are needed at different times to reduce delay and congestion in NoC. RL is used to evaluate the routing algorithms using historical data and future consequences of its current actions. An RL agent can learn about their environment and improve automatically with experience. An RL agent interacts with its environment over a discrete set of time steps. At each step, the agent senses the current state of its environment, and executes an action. This results in a change in the state of the environment (which the agent can sense in the next time step) and produces an immediate reward. The agent's goal is to maximize its long-term cumulative reward by learning an optimal policy that maps states to actions. To maximize NoC performance, we reward the agent for minimizing the latency at each time step. Based on the congestion scenario, the RL agent chooses the best routing algorithm proactively to route packets (to avoid congestion), instead of choosing different routing algorithms for different routers in NoC. Reactive techniques may not produce the rerouting solution within a required time and so loses the opportunity to avoid congestion. Additionally, the use of routing algorithm at a time reduces the possibility of deadlock in NoC.

A. Components of Reinforcement Learning Agent

RL has following four components in its model: environment, state, action, and reward.

Environment. Environment of the RL model consists of the NoC routers, links and processing cores. The environment generates the reward in terms of NoC performance for the taken NoC routing decision. **State**. NoC statistics are used as the state of the RL mode. Following features are used in the state vector for taking routing decisions in NoC: simulation seconds, received flits, and average latency. Average latency is the sum of queuing latency (at buffers) and network (routing+links) latency. **Action**. The RL agent selects a routing algorithm from the available routing algorithms for routing packets to avoid congestion for NoC performance improvement. Following routing algorithms are used: XY-routing, random-oblivious routing, adaptive west-first routing. **Reward**. Reward is formulated to minimize NoC latency to maximizing NoC performance, as shown in Eq. 1. The reward determines how good the taken action is, and the RL algorithm is designed to maximize the long-term reward.

$$reward = -average\_packet\_latency, \tag{1}$$

where, average_packet_latency is obtained from the NoC statistics.

We have three RL algorithms [45] in our work for performance evaluation: (i) **Q-learning**: learn an action-utility function Q(s,a) that tells us the value of doing action a in state s and chooses the best value, as shown in eq. 2. (ii) **Sarsa** (State-Action-Reward-State-Action): Sarsa learns action values relative to the policy it follows, as shown in eq. 3. Sarsa uses ϵ-greedy policy, where some of the actions are chosen at random. Now, for a greedy agent that always takes the action with best Q-value, the two algorithms are identical. When exploration is happening, however, sarsa and Q-learning differ significantly. Because Q-learning uses the best Q-value, it pays no attention to the actual policy being followed—it is an off-policy learning algorithm, whereas Sarsa is an on-policy algorithm (iii) **Expected sarsa** (esarsa): esarsa is like Q-learning but instead of taking the maximum over next state-action pairs, it uses the expected value, taking into account how likely each action is under the current policy, as shown in Eq. 4. esarsa is more computationally expensive than sarsa but it eliminates the variance as it considers all the actions.

$$\begin{aligned} Q(s_t, a_t) = (1-\alpha)\cdot Q(s_t, a_t) + \\ \alpha \cdot (r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a)) \end{aligned} \tag{2}$$

$$\begin{aligned} Q(s_t, a_t) = (1-\alpha)\cdot Q(s_t, a_t) + \\ \alpha \cdot (r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1})) \end{aligned} \tag{3}$$

$$\begin{aligned} Q(s_t, a_t) = (1-\alpha)\cdot Q(s_t, a_t) + \\ \alpha \cdot (r_{t+1} + \gamma \cdot E_\pi[Q(s_{t+1}, a_{t+1})]) \end{aligned} \tag{4}$$

where, t is the time step. r_t is the reward observed for the current state s_t. The learning rate α determines the importance of new experience compared to past experience. The discount factor γ determines the importance of future rewards. Lower discount rate gives more importance to current rewards, where a factor of 0 only considers current rewards.

IV. Simulation Results

Real system experiments are carried out using garnet 2.0 [3], a cycle accurate network simulator, integrated within gem5 [7] multi-core platform to evaluate the NoC performance. We simulated $8X8$ 2D-Mesh NoC topology with 64 cores, where a router is connected with each core. The simulation cycle is set to $20,000$, and we evaluate our RL solutions using uniform-random traffic pattern (which yields high-radix traffic) of synthetic traffic patterns. We observed that with low-radix traffic patterns, such as transpose and tornado, there is no significant change in NoC performance. In our experiments, we selected injection rates from 0.05 to 0.40 with an incremental step of 0.05. The injection rate is in units of packets/node/cycle. For example, injection rate of 0.4 means a cpu generates a packet in each cycle with probability of 0.4. Other NoC configuration includes interconnect frequency of 1GHz, 4 Virtual Channels (VCs), 4 buffers per VC, VC flow control, and flit-width of 128 bit. We have integrated a single RL agent for whole NoC. The centralized RL agent looks at the statistics output from gem5 to observe the states and reward to make the decision for a routing algorithm. The runtime computation for each experiment is the multiplication of the episodes and injection rates. For example, for 50 episodes and for range of injection rates from 0.05 to 0.4, each gem5 run takes around 1-2 minutes for each episode and injection rate combination in a 64-core NoC configuration.

For a given multi-core system where traffic patterns change significantly over the time, the question is how an RL agent can be able to quickly adapt to the runtime changes in application demands and/or changes in computation and communication resources in NoC to avoid network congestion. In our approach, we simulated three RL algorithms (sarsa, expected sarsa (esarsa), and Q-learning (QL)), which represent our RL agents as described in [45]. We have integrated OpenAI Gym benchmark suite [10] in gem5/gatnet for developing RL algorithms. Learning rate (α) and discount factor (γ) are set to 0.01 and 0.9, respectively. Our proposed RL models takes the current state information from gem5 output file, which includes flits received, average packet latency, and simulation time. Then an RL model selects a routing algorithm as an action for the next run based on the ϵ-greedy probability function and cumulative trained reward value. After that action, Q-values are updated in the Q-table based on the reward values generated from the current routing algorithm selection.

In an RL algorithm, the reward function is defined to maximize the negative latency or minimize the latency. The reward value is fed back to the RL model at each training episode so that the RL agent can learn the positive or negative affect of its actions. In Fig. 2, we can see that the reward values have converged and stabilized after episode 40 (that means our agent doesn't have to train for long time). We evaluated our RL models for the entire range of state transition from 0.05 to 0.40 for each episode.

As shown in Fig. 3, the overall performance slightly improves by our RL enabled solutions. We compare the latency of NoCs driven by our RL models against that of fixed routing algorithms in various injection rates as in Fig. 3. We observe that the west-first routing algorithm has a significant increase in latency after the injection rate of 0.3. The reason for increased latency in west-first routing is that west-first routing allows non-minimal paths from one node to another to avoid congestion. Since our RL models leverage the reward function with latency optimization target, the proposed models are trained to minimize latency values, which results in better NoC performance. That's why our proposed RL models are efficient for any kind of traffic demands as it can find a suitable routing algorithm to avoid congestion and to improve NoC performance.

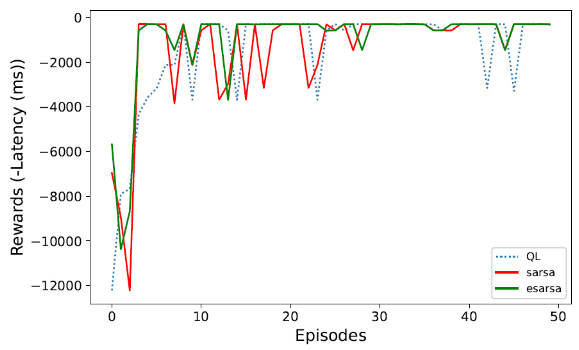

Fig. 2: Rewards over the training episodes for RL algorithms.

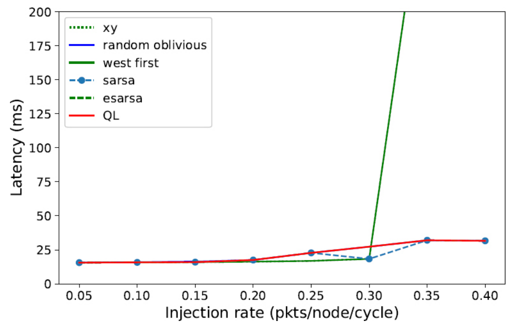

Fig. 3: Latency comparison under Synthetic Benchmarks in 64-core NoC

V. Conclusions

We have proposed a dynamic NoC routing prediction technique for congestion avoidance for high-performance NoC. Our proposed reinforcement learning enabled approach can predict the possibility of congestion and can route the packets using global network information. Different routing algorithms are used for packet rerouting depending on traffic demands for congestion avoidance and performance improvement at runtime. Simulation results under real systems shows that the proposed approach reduces latency while providing robust NoC performance for heterogeneous and changing traffic patterns.

References

[1] Cerebras Wafer Scale Engine, https://www.cerebras.net/product/.

[2] Nvidia Tesla Gpu, https://www.nvidia.com/en-us/data-center/v100/.

[3] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *Proc. ISPASS*, pages 33–42, April 2009.

[4] G. Ascia, V. Catania, M. Palesi, and D. Patti. Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. *IEEE Transactions on Computers*, 57(6):809–820, 2008.

[5] Y. Bai, V. W. Lee, and E. Ipek. Voltage regulator efficiency aware power management. In *Proc. ASPLOS*, pages 825–838, April 2017.

[6] K. Bergman et al. Exascale computing study: Technology challenges in achieving exascale systems, 2008.

[7] N. Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[8] R. Bitirgen, E. Ipek, and J. F. Martinez. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In *Proc. MICRO*, pages 318–329, 2008.

[9] B. Bohnenstiehl et al. Kilocore: A 32-nm 1000-processor computational array. *IEEE Journal of Solid-State Circuits*, 52(4):891–902, April 2017.

[10] G. Brockman et al. Openai gym. *CoRR*, abs/1606.01540, 2016.

[11] Z. Chen and D. Marculescu. Distributed reinforcement learning for power limited many-core system performance optimization. In *Proc. DATE*, pages 1521–1526, March 2015.

[12] S. M. P. D., H. Yu, H. Huang, and D. Xu. A q-learning based self-adaptive i/o communication for 2.5d integrated many-core microprocessor and memory. *IEEE Transactions on Computers*, 65(4):1185–1196, 2016.

[13] W. J. Dally and B. P. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[14] S. Das, J. R. Doppa, D. H. Kim, P. P. Pande, and K. Chakrabarty. Optimizing 3d noc design for energy efficiency: A machine learning approach. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 705–712, Nov 2015.

[15] G. Dhiman and T. Rosing. System-level power management using online learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(5):676–689, May 2009.

[16] Y. Ding, N. Mishra, and H. Hoffmann. Generative and multi-phase learning for computer systems optimization. In *Proc. ISCA*, ISCA '19, page 39–52, New York, NY, USA, 2019. Association for Computing Machinery.

[17] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, 1993.

[18] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055–1067, 1995.

[19] M. Ebrahimi et al. Haraq: Congestion-aware learning model for highly adaptive routing algorithm in on-chip networks. In *Proc. NOCS*, pages 19–26, 2012.

[20] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, and P. Liljeberg. Adaptive reinforcement learning method for networks-on-chip. *Proc. International Conference on Embedded Computer Systems (SAMOS)*, pages 236–243, 2012.

[21] Q. Fettes et al. Dynamic voltage and frequency scaling in nocs with supervised and reinforcement learning techniques. *IEEE Transactions on Computers*, 68(3):375–389, March 2019.

[22] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729–738, 2000.

[23] P. Gratz, B. Grot, and S. W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *Proc. HPCA*, pages 203–214, 2008.

[24] E. Ipek, O. Mutlu, J. F. Martínez, and R. Caruana. Self-optimizing memory controllers: A reinforcement learning approach. In *Proc. ISCA*, pages 39–50, 2008.

[25] R. Jain, P. R. Panda, and S. Subramoney. Machine learned machines: Adaptive co-optimization of caches, cores, and on-chip network. In *Proc. DATE*, pages 253–256, March 2016.

[26] D. C. Juan, H. Zhou, D. Marculescu, and X. Li. A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors. In *Proc. ASP-DAC*, pages 597–602, Jan 2012.

[27] H. Jung and M. Pedram. Supervised learning based power management for multicore processors. *IEEE Trans. TCAD*, 29(9):1395–1408, Sept 2010.

[28] E. Kakoulli, V. Soteriou, and T. Theocharides. Intelligent hotspot prediction for network-on-chip-based multicore systems. *IEEE Trans. TCAD*, 31(3):418–431, March 2012.

[29] S.-C. Kao, C.-H. H. Yang, P.-Y. Chen, X. Ma, and T. Krishna. Reinforcement learning based interconnection routing for adaptive traffic optimization. In *Proc. NOCS*, 2019.

[30] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proc. DAC*, page 559–564, 2005.

[31] R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu. Machine learning and manycore systems design: A serendipitous symbiosis. *Computer*, 51(7):66–77, July 2018.

[32] R. G. Kim et al. Imitation learning for dynamic vfi control in large-scale manycore systems. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 25(9):2458–2471, Sep. 2017.

[33] M. A. Kinsy, S. Khadka, and M. Isakov. Prenoc: Neural network based predictive routing for network-on-chip architectures. In *Proc. GLSVLSI*, pages 65–70, 2017.

[34] T. T. Le, D. Zhao, and M. Bayoumi. Efficient reconfigurable global network-on-chip designs towards heterogeneous CPU-GPU systems: An application-aware approach. In *Proc. ISVLSI*, pages 439–444, July 2017.

[35] G. D. Micheli et al. Networks on chips: From research to products. In *Proc. DAC*, pages 300–305, June 2010.

[36] G.-Y. Pan, J.-Y. Jou, and B.-C. Lai. Scalable power management using multilevel reinforcement learning for multiprocessors. *ACM Trans. Des. Autom. Electron. Syst.*, 19(4), Aug. 2014.

[37] J. Park, I. Hong, G. Kim, B. Nam, and H. Yoo. Intelligent network-on-chip with online reinforcement learning for portable hd object recognition processor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(2):476–484, 2014.

[38] Z. Qian et al. SVR-NOC: A performance analysis tool for network-on-chips using learning-based support vector regression model. In *Proc. DATE*, pages 354–357, March 2013.

[39] M. F. Reza. Deep reinforcement learning for self-configurable noc. In *Proc. SOCC*, 2020.

[40] M. F. Reza. Reinforcement learning based dynamic link configuration for energy-efficient noc. In *Proc. MWSCAS*, pages 468–473, 2020.

[41] M. F. Reza, T. Le, B. Dey, M. Bayoumi, and D. Zhao. Neuro-NoC: Energy optimization in heterogeneous many-core noc using neural networks in dark silicon era. In *Proc. ISCAS*, 2018.

[42] J. Shalf, S. Dosanjh, and J. Morrison. Exascale computing technology challenges. In *Proc. VECPAR*, pages 1–25, 2010.

[43] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles. Goal: A load-balanced adaptive routing algorithm for torus networks. *SIGARCH Comput. Archit. News*, 31(2):194–205, May 2003.

[44] A. Singh, W. J. Dally, B. Towles, and A. K. Gupta. Globally adaptive load-balanced routing on tori. *IEEE Comput. Archit. Lett.*, 3(1):2, Jan. 2004.

[45] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[46] Y. Tan, W. Liu, and Q. Qiu. Adaptive power management using reinforcement learning. In *Proc. ICCAD*, pages 461–467, 2009.

[47] S. J. Tarsa et al. Post-silicon cpu adaptation made practical using machine learning. In *PProc. ISCA*, ISCA '19, page 14–26, New York, NY, USA, 2019. Association for Computing Machinery.

[48] K. Wang, A. Louri, A. Karanth, and R. Bunescu. Intellinoc: A holistic design framework for energy-efficient and reliable on-chip communication for manycores. In *Proc. ISCA*, pages 589–600, 2019.

[49] J. 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 *Proc. HPCA*, pages 308–319, 2014.

[50] J. Yin et al. Toward more efficient noc arbitration: A deep reinforcement learning approach. In *Proc. AIDArc, held in conjunction with ISCA*, June 2018.

[51] F. Zahn, S. Lammel, and H. Fröning. Early experiences with saving energy in direct interconnection networks. In *Proc. HiPINEB*, pages 33–40, Feb 2017.

[52] H. Zheng and A. Louri. An energy-efficient network-on-chip design using reinforcement learning. In *Proc. DAC*, 2019.