IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 10 December 2019; revised 17 March 2020; accepted 15 June 2020.
Date of publication 18 June 2020; date of current version 4 March 2022.

Digital Object Identifier 10.1109/TETC.2020.3003496

Agile: A Learning-Enabled Power and Performance-Efficient Network-on-Chip Design

HAO ZHENG, (Student Member, IEEE) AND AHMED LOURI, (Fellow, IEEE)

The authors are with the Department of Electrical and Computer Engineering, George Washington University, Washington DC 20052, USA
CORRESPONDING AUTHOR: H. ZHENG (haozheng@gwu.edu)

ABSTRACT A number of techniques to achieve power-efficient Network-on-Chips (NoCs) have been proposed, two of which are power-gating and dynamic voltage and frequency scaling (DVFS). Power-gating reduces static power, and DVFS reduces dynamic power. With the goal of reducing both static and dynamic power, it is intuitive to simultaneously deploy both techniques. However, we observe that the straightforward combination of power-gating and DVFS can result in reduced power benefits and degraded performance. In this article, we uniquely combine power-gating and DVFS with the aim of maximizing the NoC power savings and improving performance. The proposed NoC design, called Agile, consists of several architectural designs and a reinforcement learning (RL) based control policy to mitigate the negative effects induced by the combined power-gating and DVFS. Specifically, a simple bypass switch is deployed to maintain network connectivity, avoiding frequently waking up the powered-off router. An optimized pipeline can simplify pipeline stages of the bypass switch to reduce network latency. Reversible link channel buffers can be dynamically allocated to where they are needed to improve throughput. In addition, the RL control policy predicts NoC traffic and decides optimal power-gating decisions, voltage/frequency levels and NoC architecture configurations at runtime. Furthermore, we explore the use of an artificial neural network (ANN) to efficiently reduce the area overhead of implementing RL. We evaluate our design using the PARSEC benchmarks suite. The full system simulation results show that the proposed design improves the overall power savings by up to 58 percent while improving the performance up to 11 percent as compared to state-of-the-art designs. The ANN-based RL implementation and bypass switch incur nominal area overhead of 5 percent, as compared to a conventional router.

INDEX TERMS Power gating, dynamic voltage and frequency scaling, network-on-chips, reinforcement learning

I. INTRODUCTION

Network-on-Chips (NoCs) [1], [2], [3] have been the standard fabric interconnecting hundreds to thousands of cores, last-level caches and memory modules in many-core systems. While NoC offers more benefits over traditional bus-based interconnect in terms of scalability and performance, it consumes a significant amount of the chip's power budget [4], [5], [6], [7], [8]. The power problem is exacerbated by the continuation of technology scaling and therefore, power-efficient NoC designs [9], [10], [11], [12], [13], [14], [15], [16], [17] are of paramount importance.

To maximize NoC power savings, multiple techniques are inevitably deployed in an integrated manner. For example, power-gating is an effective technique to reduce static power [18], [19], [20], [21], and Dynamic voltage and frequency scaling (DVFS) is often deployed to reduce dynamic power [22], [23], [24], [25], [26]. It is intuitive to take advantage of power-gating and DVFS to simultaneously reduce both static and dynamic power. However, the straightforward combination of power-gating and DVFS can lead to severe performance degradation and diminished power savings.

The performance degradation results from the wake-up latency of power-gating [18], [27], [28] and reduced voltage/frequency levels of DVFS [24], [26]. Specifically, the powered-off router needs to take a number of cycles to resume its full activity, called wake-up latency. The latency overhead is compounded by the intermittent NoC traffic which frequently wakes up the powered-off router [18], [27]. Furthermore, DVFS reduces the operating speed of router which makes

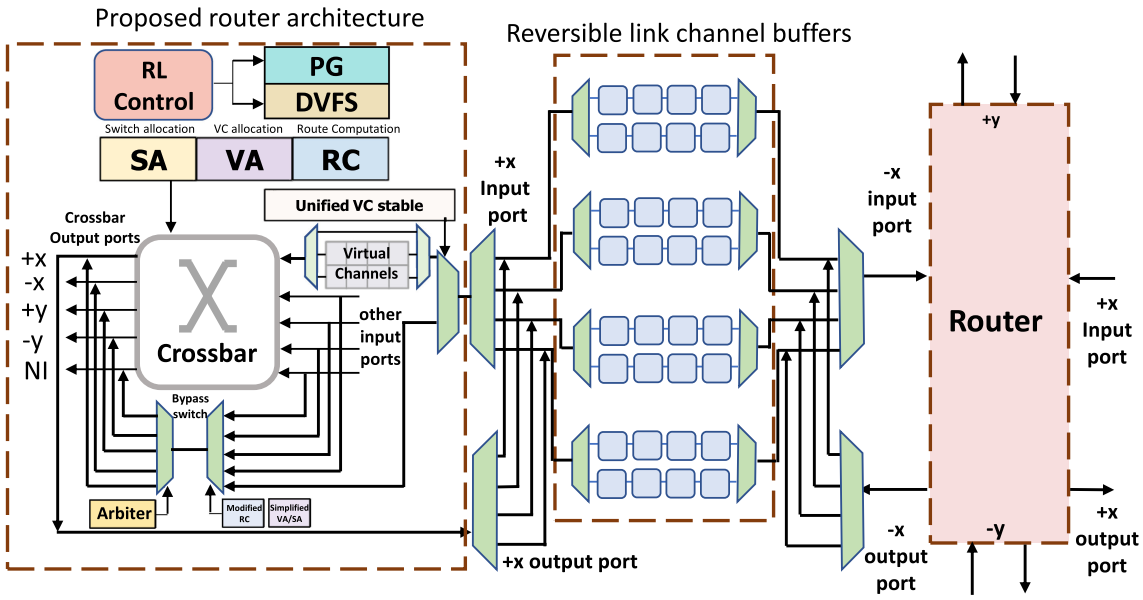

FIGURE 1. **Agile NoC architecture consisting of a conventional router, a bypass switch, reversible links, and RL-based control policy.**

the router process fewer flits in a given time, thus affecting the network throughput. The combined use of power-gating and DVFS could aggregate the performance penalties, degrading system performance.

Moreover, the dynamic interaction between power-gating and DVFS can jeopardize the efficiency of each power saving technique, reducing the overall power savings. For example, as the NoC traffic often varies in time and space, such varying traffic often results in inadequate DVFS decisions. The inadequate DVFS decisions could have negative impact on network throughput which turns to network saturation. In this case, the idle cycles between flits are significantly reduced. Recall that power-gating decision is made on the idle cycles between flits. Consequently, the DVFS can hurt the efficiency of power-gating, reducing the static power savings. We use full system simulation to analyze the adverse consequence of a straightforward combination of power-gating and DVFS using PARSEC benchmark suite. Simulation results show that the straightforward combined techniques can result in 43 percent increased latency and 26 percent reduced static power savings on average, as compared to only using power-gating.

In this paper, we propose Agile, a reinforcement learning (RL) based NOC design that uniquely combines the power-gating and DVFS with the goal of maximizing power savings and improving performance. Several architectural designs and a per-router RL based control policy are used to mitigate the negative effects induced by the simultaneous deployment of power-gating and DVFS. Additionally, we explore the use of artificial neural network (ANN) to reduce the hardware costs of implementing the RL on a per-router basis. The major contributions are the following:

Agile NoC architecture that includes (1) a simple bypass switch that processes intermittent NoC traffic when the router is powered-off, and thus avoiding frequent router wake-ups, (2) an optimized router pipeline that simplifies pipeline stages when using the bypass switch, and thus reducing the network latency, and (3) reversible link channel buffers that can be used to store intermittent flits when router is powered-off, and dynamically allocate the bandwidth and buffers where they are needed most to increase the throughput.

RL-based control policy that predicts NoC traffic and selects power-gating decisions, voltage/frequency levels, and NoC architecture configurations, thereby providing optimal NoC performance and power savings at runtime.

ANN-based RL implementation that can efficiently minimize the area overhead required by RL by storing a few weights instead of the state-action table.

Our simulation shows that the proposed design improves the static power savings by 40 percent, dynamic power savings by 17 percent , system performance by 11 percent, as compared to state-of-the-art techniques. The ANN-based RL implementation reduces the area overhead by 67 percent as compared with a table-based RL implementation.

II. PROPOSED AGILE NOC ARCHITECTURE

In this section, we describe the details of proposed Agile NoC architecture. The Agile NoC consists of a conventional router, a bypass switch, reversible links and RL control as shown in Figure 1. Specifically, we describe the conventional router in Section II-A, the bypass switch in Section II-B, reversible links in Section II-C. Furthermore, we discuss the deadlock avoidance in Section II-D. The RL control is detailed in Section III.

A. CONVENTIONAL ROUTER

The conventional router consists of virtual channels (VCs) for storing arriving flits, crossbar for switching flits from input ports to output ports, route computation (RC) for calculating flit route, virtual channel allocation (VA) for assigning

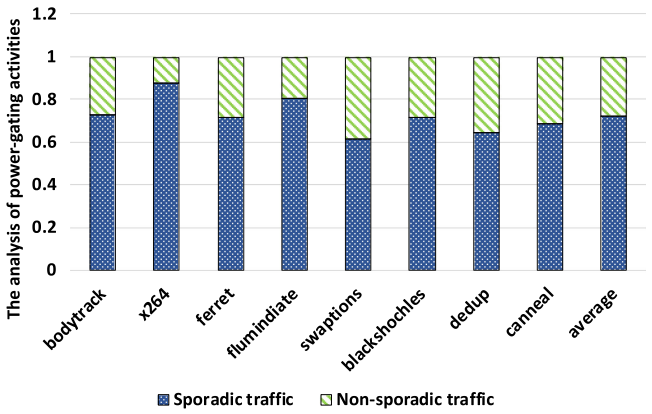

FIGURE 2. The fraction of power-gating switching activities caused by the traffic where flits are with gaps of more than four cycles.

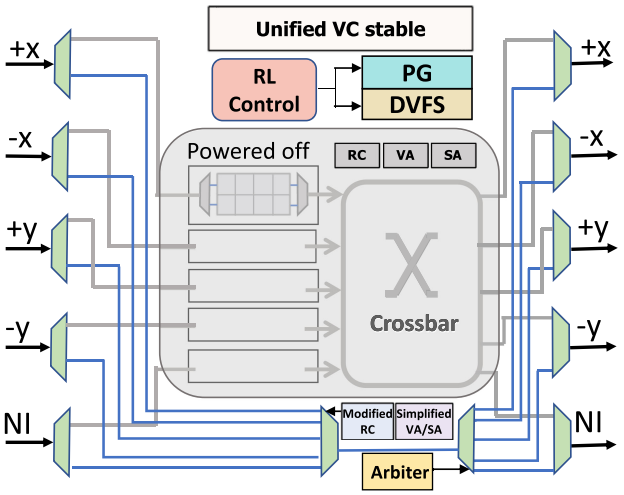

FIGURE 3. Proposed bypass switch consisting of muxes, a 5:1 arbiter, demuxes, and a unified VC state table.

virtual channels and flow control, switch allocation (SA) for allocating the available out ports to the flits requested, and switch traversing (ST) for flits passing through the crossbar [2]. In this paper, a flit passes through the router pipeline in four cycles.

B. BYPASS SWITCH DESIGN

1) BYPASS SWITCH ARCHITECTURE

The bypass switch provides an alternative datapath to process intermittent NoC traffic when the router is powered-off, avoiding the large wake-up latency. This design is motivated by our study of NoC traffic behavior, as illustrated in Figure 2. While significant research has shown that the intermittent NoC traffic behavior results from inconstant cache coherence activities such as request, reply and invalidation [29], [30], we further study the temporal behavior of the intermittent NoC traffic in order to remedy the prohibitive wake-up latency. Specifically, we define two categories of NoC traffic, namely, sporadic traffic and non-sporadic traffic. The sporadic traffic consists of the flits that have gaps of more than four cycles, while the non-sporadic traffic refers to the flits that have gaps of fewer than four cycles. We observe that approximately 73 percent of router wake-ups are caused by the sporadic traffic. As the router has a pipeline stage of four cycles, in this case, the pipelined router is under-utilized. This indicates that a simple unpipelined switch is sufficient to process the sporadic traffic when the router is powered-off.

Figure 3 depicts the proposed bypass switch. The bypass switch consists of multiplexers (MUXs), demultiplexers (DEMUXs), an arbiter, and a unified VC state table. As compared to other NoC bypass techniques [11], [31], [32], the proposed bypass switch can process sporadic traffic from any input ports to desired output ports. Specifically, each input port can store one flit in the reversible link channel buffer (Section II-C1), because the majority of VCs are under-utilized when flits are with large gaps of cycles. The bypass switch consists of a 5:1 mux and an 1:5 demux, which allows for a single flit switching. The arbiter associated with switch arbitrates the flits in a round robin manner. When the conventional router is powered-off, the unified VC table is powered-on and used to record VC information, which is detailed in Section II-C2.

2) BYPASS SWITCH PIPELINE OPTIMIZATION

To reduce network latency, we eliminate redundant delays of SA and VA stages when using the bypass switch. In the conventional router, the SA and VA stages take a significant portion of router critical delays [33], [34] due to the complexity of handling a large number of VCs in the arbitration [33]. However, in the case of sporadic traffic, the bypass switch only stores one flit at each input port. This implies that the complicated arbiters designed for the conventional router are not necessary for the bypass switch. We take this opportunity to use a simple arbiter to simultaneously arbitrate the VC and output port to the request flits, thereby reducing the router critical delays.

In this work, we modify the pipeline stages of the bypass switch. The proposed bypass switch pipeline consists of a modified RC stage, a simplified VA/SA stage, and a ST stage. Specifically, the modified RC calculates the output port of the header flit, and assigns any available VC from the unified VC state table to the flit at each input port. The flit with assigned VC sends the requests to the simplified VA/SA arbiter. The simplified VA/SA will allocate the bypass switch and VC to one of the request flits. The flits that lost the arbitration will be assigned with a new VC. The proposed design, therefore, can eliminate about 46 percent of the critical delay as compared to the conventional router pipeline. The detailed critical path analysis is shown in Section V-C1.

C. REVERSIBLE LINK CHANNEL BUFFER

1) REVERSIBLE LINK CHANNEL BUFFER ARCHITECTURE

The goal of reversible link channel buffer is to store the intermittent flits when the router is powered-off, and allocate link bandwidth to where they are needed most to increase network throughput. Specifically, when the conventional router

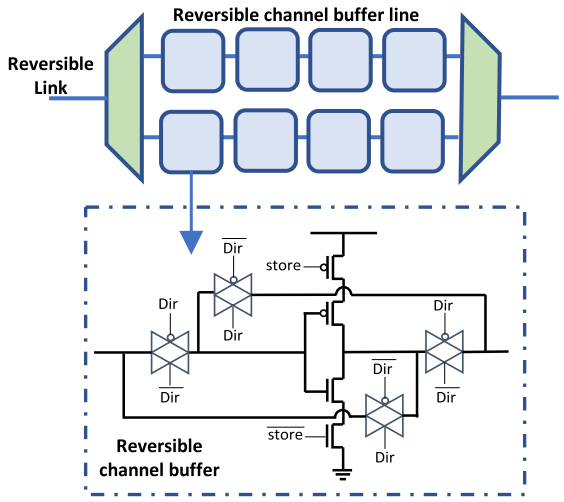

FIGURE 4. Reversible link channel buffer.

is powered-off, VCs are not available for storing the intermittent flits. This requires additional buffers to store the intermittent flits [28], [35], [36], [37]. Moreover, the reduced operating speed can have negative impacts on the network throughput. To efficiently increase the network throughput, previous work [38], [39] dynamically allocates the link bandwidth to where they are needed at most. Consequently, we deploy the reversible link channel buffers [38] to store intermittent flits and dynamically allocate the link bandwidth.

Figure 4 shows a reversible link channel buffer [38], which consists of an inverter, two additional transistors, and four transmission gates. Specifically, each channel buffer line consists of several reversible link channel buffers. The inverter is used to reduce the propagation delay associated with the wire. Two additional transistors are added into the repeater to function as a channel buffer [12]. When the 'store' signal is enabled, these transistors are turned off, and data can be stored in the link. Four transmission gates are used to reverse the link direction. By doing so, the link can be used for data transmission in both directions. For example, when the 'Dir' signal is enabled, it provides backward data transmission.

In this paper, each router is connected to its adjacent routers by four 64-bit links, which accounts for 128-bit link bandwidths for each direction. Each link has two link reversible channel buffer lines, and each line has four link reversible channel buffers. We use the reversible link channel buffer to store the data in the link when the router is powered-off and change the link directions where they are needed to increase network throughput.

2) IMPROVED FLOW CONTROL

We propose a unified VC state table with the goal of guaranteeing the correct flow control between routers. The unified VC state table coalesces all VC state tables of each input ports, and includes additional entries to record different power states and VC information for the following rationale.

First, as the conventional VC state table is associated with each input port, it cannot be accessed when the router is powered-off. The unavailable VC information can result in packet drop or misrouting. Consequently, we unify the VC state tables of all input ports [28], [40], and always power on the unified VC state table for recording the VC information.

Second, the conventional VC state table only includes the VC information, such as VC number and Credits [2]. Such information is not sufficient to represent the changed NoC architecture. For example, a unified VC state table requires a new index to identify the VC information among different input ports, and thus we add a new entry to indicate the input port. Moreover, the number of buffers can be changed due to the changed power state and link bandwidth. Therefore, we add additional entries to record the varying power states and link channel buffers.

D. DEADLOCK AVOIDANCE

The network deadlock can arise due to protocol and routing issues. To prevent the protocol deadlock, we deploy virtual networks (VNs) at both link channel buffer and input ports. In this case, the reply and request packets are stored into different VNs, avoiding the reply-request dependency. The routing deadlock could happen in case of circular channel dependency or misrouting induced by improper link reverse. We use dimensional ordered routing algorithm that places turn restriction to prevent any circular channel dependency in the mesh topology. While the circular channel dependence can be avoided, the reversed link direction may misroute the packets to a wrong direction. To address this issue, we switch the link direction until the packets are properly drained to the VCs of input ports.

III. PROPOSED RL-BASED CONTROL POLICY

It is challenging to simultaneously manage different power-saving techniques and NoC configurations, as it requires to monitor a large number of system parameters, predict the abrupt NoC traffic, and handle the immense design space. The problem is compounded by the dynamic interaction between different techniques. To this end, we explore the use of RL to automate an optimal control policy.

In this section, we describe the details of proposed RL-based control policy. Specifically, we begin with the RL basics in Section III-A. Furthermore, we describe the actions of the RL policy in Section III-B, the state space and state-action table in Sections III-C and III-D, the reward function in Section III-E, and the ANN design for reducing RL implementation overheads in Section III-F.

A. REINFORCEMENT LEARNING BASICS

Reinforcement learning (RL) [41], [42] is a machine learning approach in which an *agent* acts as a learner and decision maker by interacting with system *environment*. Figure 5 illustrates the dynamic interaction between the RL agent and the system environment: ① The agent selects an action a_t from a set of actions, $\mathcal{A} = \{1, \ldots, K\}$, at time step t; ② the selected action is applied to the environment and influences current *state* s_t and *rewards* r_t; ③ such influence results in a

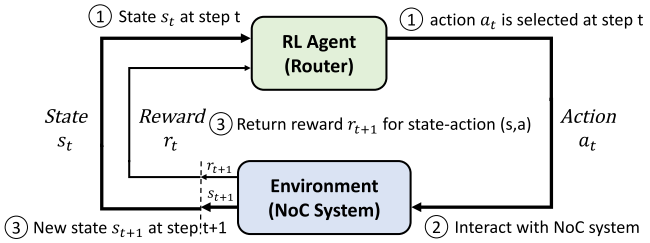

FIGURE 5. The agent-environment interaction in RL.

transition to new state and reward, s_{t+1} and r_{t+1}, at the next time step $t + 1$.

The objective of action selection is to maximize the long-term total rewards $\mathcal{R}$, which is the cumulative sum of all future rewards $(r_{t+1}, r_{t+2}, ...)$, as illustrated in Equation 1. The future rewards are discounted by a factor of γ $(0 \leq \gamma \leq 1)$, called the *discount factor*. The discount factor determines the weight of future rewards. For example, the agent becomes near-sighted and only considers current rewards when γ approaches 0.

$$\mathcal{R} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \tag{1}$$

The RL agent explores various actions and eventually evolves an optimal RL policy π over many time steps, where an action-value function maps the reward of each action to different state. For example, tabular Q-learning is one of the RL algorithms that records the action-value function as a Q-value table. The Q-value table is initialized with random values for all possible (s,a) pairs. At each time step, the RL agent selects the action based on the Q-value, and updates the action-value table entry Q(s,a) using Equation (2) based on action a, reward r, and new state s'. Over many time steps, all actions and states are explored which eventually automates an optimal Q-table.

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma maxQ(s',a) - Q(s,a)], \tag{2}$$

where α is the learning rate, γ is the discount factor, and $maxQ(s',a)$ is the maximum Q value over all possible actions in state s'.

B. ACTIONS

In RL, actions are taken to optimize the system performance. In this work, we incorporate different power-gating decisions, voltage/frequency levels, and NoC configurations to improve NoC power-efficiency and performance. The power-gating and voltage/frequency levels are dynamically selected to reduce both static and dynamic power, however, inevitably affect the NoC latency and throughput. Although previous work [43], [44], [45] has studied the impacts of different NoC latency and throughput on application performance, we come to the similar conclusions but interpret them differently. In our opinion, the network latency is critical to sporadic traffic, as the sporadic traffic cannot utilize the router pipeline to overlap the latency. As a result, the performance improvement can only be achieved by reducing the latency that each flit traverses the router. On the other hand, the bandwidth is critical to non-sporadic traffic because the non-sporadic traffic can overlap the latency using the router pipeline. In this case, reducing the router latency cannot effectively improve the pipeline throughput, and thereby bandwidth is becoming important. To this end, we select the optimal NoC architecture to reduce network latency or improve network throughput according to the traffic behavior.

Specifically, the actions consist of five different operation modes. Each operation mode consists of different power-gating decisions, voltage/frequency levels, and architecture configurations. The power-gating decisions can disconnect the conventional router from the power supply to reduce static power. The voltage/frequency levels are categorized to three modes, namely, high, middle and low. The agile NoC architecture can be configured at runtime to mitigate the negative effects of power-gating and DVFS. The operation modes are as below:

1) *Mode-0*: The conventional router is powered-off, and the voltage/frequency level is set at the low mode. In this case, the power-gating and DVFS can reduce dynamic and static power. The bypass switch is enabled to process the intermittent flits that cannot utilize the conventional router pipeline, and the optimized pipeline is used to reduce network latency. The link bandwidths are equally allocated to each direction, and the reversible link channel buffer can store one intermittent flit in each link at a time.
2) *Mode-1*: As compared to mode-0, we increase the voltage/frequency level to the middle mode, and thus providing higher operating speed but less dynamic power savings. The increased operating speed is used to reduce the latency caused by the power-gating and DVFS. The bypass switch is enabled to process the intermittent flits, and reversible link channel buffer is used to store these flits.
3) *Mode-2*: This mode is used to extend the static power savings at the expense of dynamic power savings. In this case, the voltage/frequency levels are increased to the high mode. Therefore, a high speed bypass switch can handle more intensive NoC traffic at the expense of dynamic power savings, as compared to mode 0-1.
4) *Mode-3*: This mode is designed to provide more router throughput, as compared to mode 0-2. The conventional router is used to increase router storage (i.g., VCs) and provide a robust crossbar, and thus increasing the router throughput. Since the increased number of VCs requires complicated VA and SA which increases the critical delay, the pipeline needs to be resumed to 4-stage. The voltage and frequency level is set as middle mode to save dynamic power. To compensate the reduced throughput caused by DVFS, we use a dynamic bandwidth allocation scheme [38] to dynamically allocate the link bandwidth where they are needed at most, and the bypass switch is allocated to the input port that has the most of traffic in the past time epoch.

Category	State Attributes	Description
Cache Related Metrics	1. L1D cache miss	the number of L1 data cache misses
	2. L1I cache miss	the number of L1 instruction cache misses
	3. L2 cache miss	the number of L2 cache misses
Network Related Metrics	4. +X buffer utilization	the buffer utilization of +X input port
	5. –X buffer utilization	the buffer utilization of -X input port
	6. +Y buffer utilization	the buffer utilization of +Y input port
	7. -Y buffer utilization	the buffer utilization of –Y input port
	8. Local port buffer utilization	the buffer utilization of local port
	9. Router throughput	the number of flits received per epoch
Message Information	10. Response flits	the number of response flits received
	11. Request flits	the number of request flits received
PG and DVFS Interaction	12. PG efficiency	the efficiency of power-gating ($T_{power\text{-}off}/T_{epoch}$)

FIGURE 6. The state attributes used in RL.

5) *Mode-4*: This mode is used to provide highest router throughput by increasing the voltage/frequency levels to high mode as compared to mode-3.

C. STATE SPACE

As the RL agent makes a decision by observing the system parameters, it is critical to make the router aware of more system parameters to increase the prediction accuracy. In RL, the parameters are recorded as a state vector s. In this paper, each state has 12 system metrics related to cache and network performance, as shown in Figure 6.

- *Cache Related Metrics:* As the L1 cache is attached to the cores, the activities of the L1 cache are correlated to the computation intensity. We, therefore, use state attributes 1-2 to indicate the activities of a local core. Moreover, as L2 caches are shared and distributed across the chip, they communicate through the NoC. These communications are good indications of overall NoC global behavior.
- *Network Related Metrics:* To make the router aware of different NoC traffic, we use state attributes 4-9 to represent the traffic intensity at each port. Moreover, flits are often categorized into two classes called response and request (state attributes 10-11) in NoCs. Response flits often have priority over the request flits due to message dependence. State attribute 12 is the PG efficiency of a router, indicating the efficiency of the PG for different DVFS decisions.

D. STATE-ACTION TABLE

RL agent selects actions according to the Q-values of a given state. Q-values for any state-action pair are recorded independently in a per-router based state-action table. An example is shown in Figure 7. The RL agent observes states and records them as a vector $<0,2,2,...>$ in a given time t, where each element of the vector represents a specific state. Because some state attributes are continuous numbers, they could lead to infinite state space and infeasible Q-learning converging time. Therefore, these continuous numbers are discretized into finite bins. For example, the RL agent monitors that the router was powered-off for 10K cycles in the past 10K-cycle epoch, which means its PG efficiency is 1 (powered-off time/epoch time). In this paper, each state attribute has a discrete set of 5 bins, $\{0,1,2,3,4\}$.

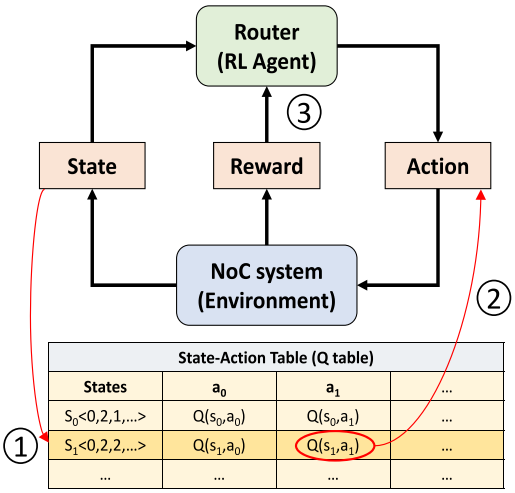

FIGURE 7. The state-action table of RL.

The entire RL process is shown in Figure 7: ① The observed state vector $<0,2,2,...>$ is used to query the table where S_1 matches the state $<0,2,2,...>$. In the matched entry of Q table, $Q(s_1, a_1)$ is of the maximum value, and thus action a_1 is selected. In this work, the ε-greedy policy is applied to explore additional action space, in which a random action could be selected with a small probability of ε. ② Upon applying the selection action, the system will transition to state s' in the next time step t'. ① By following the Equation (2), the Q-value is a sum of past rewards and immediate reward. The immediate reward results from the action a_1 and the discounted future reward $\gamma maxQ(s', a')$.

E. REWARD FUNCTION

The RL agent uses a reward function to evaluate how beneficial it is to take a specific *action* for a given *state*. Typically, choosing an action with a higher reward function tends to result in better system performance (e.g., power savings). In other words, the reward function instructs the RL agent to maximize the long-term reward, which in our case implies maximizing the overall power savings and minimizing the network latency. Thus, we design the reward function for the router as follows:

$$Reward = -log(latency) - log(power). \tag{3}$$

Power refers to the NoC static power, dynamic power, and power-gating overhead of a given router. Latency is the average read miss latency measured in the miss status holding register. Both static and dynamic power are calculated by DSENT model, where static power is estimated by using the PG efficiency. Power-gating overhead is the product of number of power-gating activities and switching overhead. The dynamic power is calculated by the number of buffer writes, crossbar, VA and SA activities in each time epoch. To calculate this average read cache miss latency, we record the time

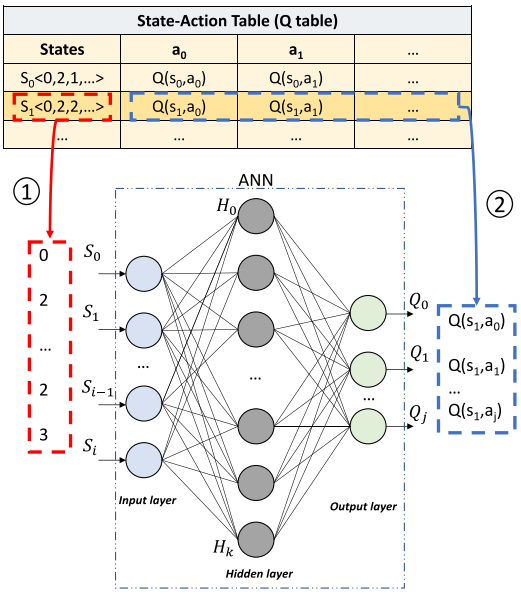

FIGURE 8. The ANN calculates the Q-values instead of using a state-action table.

difference between the issued time and the completed time of each MSHR entry.

F. ANN FOR REDUCING RL IMPLEMENTATION OVERHEAD

In RL, each state vector consists of a number of states, as discussed in Section III-C. When the RL agent observes any new state-action pair, it creates a new entry in the state-action table to record its actions and associated Q-values. Although we discretize the parameters into finite bins for a smaller table size, our simulation results show that the state-action table still requires over 10 percent router area to store all state-action pairs. To address this problem, we replace the state-action table with an offline-trained ANN to impose lower hardware costs. The ANN calculates the state-action table instead of storing the entire state-action table in the router, thus eliminating the storage space for state-action pairs. It should be noted that although prior work [46] deploys a deep convolutional neural network to approximate the state-action table, the complexity of convolutional neural network is inapplicable to reduce hardware costs.

ANN Architecture. The proposed ANN consists of 3 layers: an input layer, a hidden layer and an output layer. Each layer consists of multiple neurons. The Sigmoid and Relu functions are used for the hidden and output layers, respectively. The sizes of the input and output layers depend on the designs of the state and action space. A more detailed discussion of the hidden layer size will be discussed in Section VI-B.

Training and Inference Details. In offline training, the ANN takes each state-action entry as a training sample. The state vector is used as the input of the ANN, and Q values are used as the desired output values. Note that the input values are normalized in the range of 0 to 1 due to the nonlinearity of the sigmoid function. We use the mean square error function to calculate the error between the output of the ANN and the desired values, and then we use mini-batch gradient descent to back propagate this error to the hidden layer to tune the weights. The batch size is set to 100 in this work. For the offline approach, we use 0.001 as the learning rate because the accuracy is more important than the speed.

TABLE 1. Key simulation parameters.

# of cores	64 out-of-order CPUs, ALPHA, 2GHz
Router	4-stage pipeline
L1 private I/D cache	64 KB, 4-way, 1 cycle, LRU
L2 shared cache	1 MB, 16-way, 6 cycle, LRU
Cache block size	64 Bytes
Buffer size	4 buffers/VC, 4 VCs/VN (Baseline)
Protocol	MESI
Memory latency	128 cycles
Topology	8×8 Mesh, XY routing

Figure 8 shows a walkthrough example of using an ANN. ① $S_1 < 0,2,2,...>$ is monitored by an RL agent, which is used as the input value of the ANN. ②The network calculates the input values layer by layer and then delivers five values to the output layer, $Q(s_1, a_0)$, $Q(s_1, a_1)$, $Q(s_1, a_2)$, $Q(s_1, a_3)$ and $Q(s_1, a_4)$. The action with the highest Q-value will be selected.

IV. EVALUATION

We evaluate the proposed architecture using full system simulation with the combination of architecture-level and circuit-level simulators. The cycle-accurate gem5 simulator [47] enhanced with GARNET [48] is used for detailed timing simulation of the memory and on-chip network. We use DSENT [49] to evaluate power consumption and use a Synopsys design compiler with the 45 nm FreePDK45 Open Cell Library [50] to evaluate the area overheads. Table 1 shows the specific parameters used in the simulation. For fair comparison, the baseline has the same number of buffers as the Agile design. We assume 4 128-bit buffers per VC and 4 VCs per VN in the baseline router. In the Agile NoC, we assume 2 128-bit buffers per VC and 4 VCs per VN for each router. Each reversible link has 8 64-bit buffers, which can store 4 128-bit flits. We analyze our framework with the PARSEC benchmark suite, in which different application characterizations, traffic load and phase behavior, are considered [51]. For example, the applications, flumindiate and ferret applications, are composed of substantial synchronization primitives, leading to frequent phase changes.

A C++-based RL algorithm is implemented with GARNET simulator to provide closed-loop simulation. Since the learning rate α, the discount rate γ, and exploration rate ϵ are hyper-parameters, we set learning rate α to 0.1, discount factor γ to 0.95, and set exploration rate ϵ to 0.1. The studies of these parameters are detailed in Section VI.

For ridge regression, we create the training sets by profiling 10 million cycles of each application. A python-based machine learning tool [52] is used to train the ridge

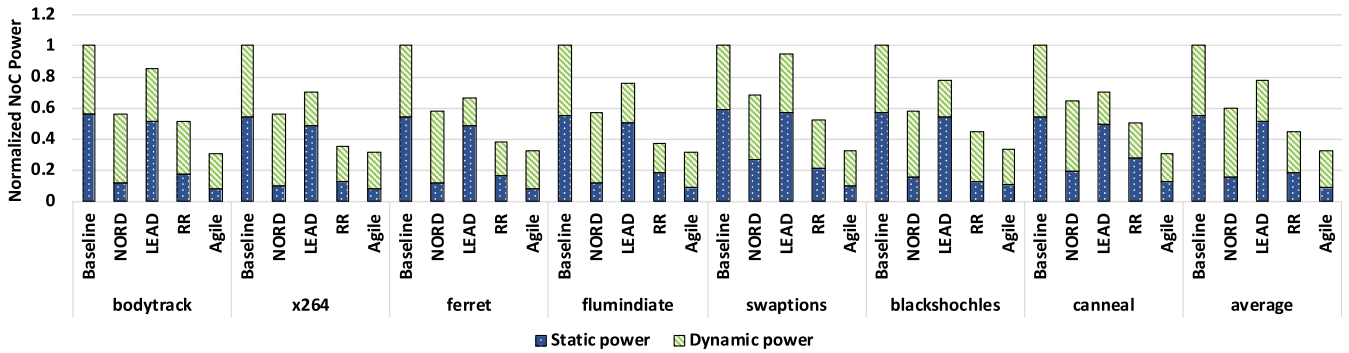

FIGURE 9. The overall router power consumption comparison, normalized to the baseline.

regression model, and we feed the trained results to GARNET for the closed-loop simulation.

We assume voltage levels from 0.6 V to 1 V and frequencies from 1 GHz to 2 GHz. The voltage and frequency levels are 2GHz/1V, 1.5GHz/0.8V, and 1GHz/0.6V for high, middle and low modes respectively. Upon receiving a decision from the RL agent, the DVFS controller selects the appropriate V/F for saving the router's dynamic power every 10 K cycles. We set the voltage regulator transition latency to 100 ns [53]. The wake-up latency of power-gating are set as 8 ns [27].

The simulation framework consists of the following components:

(1) Baseline: A baseline NoC architecture is without the deployments of power-gating and DVFS.
(2) NORD [27]: A per-router-based power-gating is implemented to save static power, where an additional bypass ring is implemented to route the packets when the router is powered-off.
(3) LEAD [54]: A per-router-based DVFS is implemented to save dynamic power. A supervised learning algorithm, ridge regression (RR), is used to select the voltage and power levels.
(4) RR: The NoC can choose different power-gating decisions, voltage/frequency levels and architecture configurations. Ridge regression [54] is used to select the operation modes as described in Section 4.2. We use the same threshold as the LEAD to set the threshold for categorizing the traffic, which is used for the training and inference stages of ridge regression.
(5) Agile: The proposed design can move into different power levels and architecture configurations to maximize the power savings and improve performance. An RL-based control policy is used to select the optimal operation mode at runtime.

V. RESULTS

A. POWER CONSUMPTION ANALYSIS

1) DYNAMIC POWER CONSUMPTION ANALYSIS

Figure 9 shows the dynamic power consumption analysis. The dynamic power savings mainly result from the reduced voltage/frequency levels. As the NORD only integrates power-gating, it has the maximum dynamic power which is close to the baseline. LEAD and RR have similar dynamic power reductions as compared to the baseline. The RR has 5 percent more dynamic power consumption as compared to the LEAD. The agile has the highest dynamic power savings on average, which has up to 17 percent more dynamic power savings as compared to supervised learning based techniques (i.e., ridge regression). However, agile performs worse in some applications as compared to RR, such as x264, ferret, and flumindiate. These applications sometimes have frequent phase changes, leading to bursty NoC traffic. The RR is optimized for dynamic power savings and could fail to identify the transient traffic burst because it only monitors the average network utilization of each time epoch. As a result, the RR selects reduced voltage/frequency for reducing dynamic power. For agile, it is optimized for both static and dynamic power while considering a number of system parameters. In this case, it increases the static power savings at the expense of dynamic power savings with the aim of maximizing the overall power savings. Specifically, a high speed switch can handle more intensive NoC traffic to extend the powered-off time of conventional router.

2) STATIC POWER CONSUMPTION ANALYSIS

Figure 9 shows the static power consumption analysis. The static power savings result from the deployment of power-gating. LEAD is only integrated with DVFS and, therefore, has the least static power savings which are similar to that of baseline. We normalize the static power to baseline. NORD with a bypass ring provides an alternative way to route the intermittent flits, extending the sleep time, and saving 76 percent more static power than LEAD. RR has an average of 23 percent more static power savings than LEAD. However, we observe that in some applications (e.g., x264, ferret, and flumindiate), RR has worse performance than NORD because we proactively enable power-gating rather than reactively using power-gating, as in NORD. RR performs worse under more intensive traffic. Agile has 40 percent more static power savings than NORD because of the simplified pipeline stages and better network scalability.

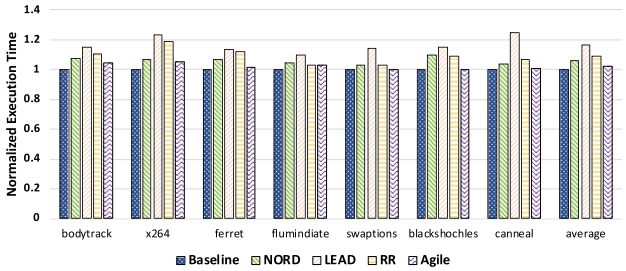

FIGURE 10. Overall execution time comparison, normalized to the baseline.

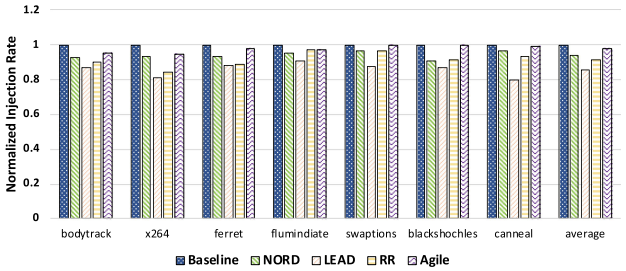

FIGURE 12. Injection rate comparison, normalized to the baseline.

3) OVERALL POWER CONSUMPTION ANALYSIS

Figure 9 shows the overall power consumption. The agile has an average of 68, 58, 46, and 27 percent more overall power savings compared to baseline, LEAD, NORD, and RR. As static power takes a significant portion of the overall power consumption, LEAD, only with DVFS, has the least overall power savings. NORD, only with power-gating, cannot reduce the dynamic power and, therefore, has the second least overall power savings. RR and Agile have both power-gating and DVFS, thus reducing both static and dynamic power consumption. However, agile yields better overall power savings due to more accurate traffic prediction and better trade-off between dynamic and static power savings.

B. PERFORMANCE ANALYSIS

1) EXECUTION TIME ANALYSIS

Figure 10 shows the performance analysis for various techniques. Agile has 5, 7, and 11 percent more execution time reduction than NORD, RR, and LEAD. LEAD has the longest execution time, which implies that the DVFS performance loss is inevitable even with an accurate traffic prediction. RR has significant performance improvement as compared to LEAD because the bypass switch and reversible links are implemented to improve network performance. When we compare the performance between supervised learning and RL, RL has better performance over the supervised learning. This is because agile can more accurately capture the dynamic interactions between each time epoch. We note that agile only imposes 2 percent execution time overhead as compared to the baseline.

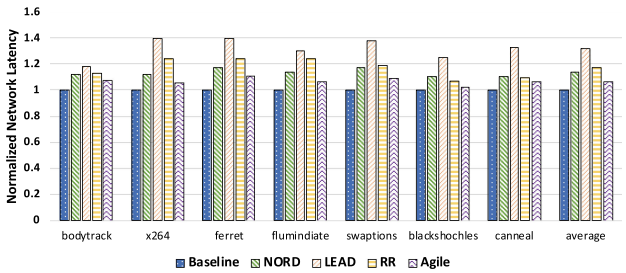

FIGURE 11. Network latency comparison, normalized to the baseline.

2) NETWORK LATENCY ANALYSIS

Figure 11 shows the network latency analysis. The execution time is often related to network latency [55], though applications have different sensitives to the network latency. This is because the higher network latency can delay requests and responses of coherence messages, and slow down the data transmission, resulting in poor system performance. Agile has 8, 11, and 23 percent network reduction compared with NORD, RR, and LEAD. LEAD has the highest network latency, which implies that the DVFS performance loss is more than the optimized power-gating technique. When comparing the performance between supervised learning and RL, RL has better performance over the supervised learning. As compared to baseline, agile only incurs 6 percent additional network latency.

3) INJECTION RATE ANALYSIS

Figure 12 shows the injection rate analysis. Agile, NORD, RR, and LEAD have 2, 7, 10, and 16 percent injection rate reductions as compared to the baseline, respectively. As the injection rate reduction mainly results from inadequate injection throughput and increased network latency. The inadequate injection throughput makes the flits congested at the injection port; and increased network latency delays the process of request-reply in cache coherence. As the agile is capable of processing the intermittent flits without delaying the request-reply and reversing the injection port to avoid flit congestion, it barely affects the injection rate even power-gating and DVFS are deployed.

C. OVERHEAD ANALYSIS

1) CRITICAL PATH ANALYSIS FOR ROUTER PIPELINES

To verify the critical delay of the proposed designs, we use Synopsys to evaluate the critical path delays of the pipeline stages with the PDK 45 nm library. In a conventional router, the RC, VA, SA, and ST stages take 0.24 ns, 0.45 ns, 0.33 ns, and 0.59 ns, respectively. The additional logic added to the RC stage incurs a 0.1 ns critical delay, and the 5:1 allocator incurs a 0.15 ns critical delay. A simple switch only incurs a 0.26 ns critical delay. The bypass route only incurs a 0.75 ns critical delay in total, which accounts for 46 percent of the critical delay of conventional routers. As a result, the critical delay of the bypass switch can fit into two cycles at 2GHz.

TABLE 2. NoC area (um^2).

	RC	VA	SA	Crossbar	Buffer	Link	RL	Total
Baseline	429	9066	4589	17806	197480	269	N/A	229639
Agile	785	9066	4773	18176	98740	82417	8353	221311

2) TIMING OVERHEAD FOR ANN CALCULATION

We define the timing overhead as the time that ANN calculates Q-values. The timing overhead depends on the number of hardware resources in the RL agent. In this paper, we assume one adder and one multiplier for the ANN, which consists of 12 neurons in the input layer, 20 neurons in the hidden layer, and 5 neurons at the output layer. With this design, the ANN calculation incurs 330 ns latency. This latency can be overlapped by a large epoch equal to 10 K cycles.

3) AREA OVERHEAD FOR IMPLEMENTING RL

We evaluate the hardware cost through Synopsis Design Vision using 45 nm technology. Table 2 shows the areas of different NoC components, including the crossbar, SA, VA, router buffers, RL implementation, and reversible link channel buffers. The baseline router consists of a crossbar of 17806 um^2, an SA of 4589 um^2, a VA of 9066 um^2, and buffers of 197480 um^2. The neural network requires an ALU (e.g., multiplier, sigmoid function) and storage. The total area of the overhead of the ANN is 8353 um^2. The modified RC, the unified VC table, the switch arbiter and the reversible links account for 83959 um^2 in total.

The bypass switch and ANN only incur about 5 percent area overhead as compared to a conventional router. The reversible link channel buffers can reduce about 17 percent area overhead as compared to register-based router buffer. In total, Agile requires 3 percent less area than baseline. The ANN-based RL implementation results in 67 percent less overhead than the table-based RL implementation by removing the SRAM storage for the large Q-table.

VI. SENSITIVITY STUDY

A. IMPACTS OF EPOCH SIZE ON SYSTEM PERFORMANCE

To study the impact of different epoch sizes on performance, we vary the epoch size from 500 to 20000 cycles. Figure 13 shows that the epoch size of 10K has the best network latency performance and energy efficiency. As the voltage regulator imposes a large transition latency of 100 ns, a smaller epoch size could frequently stall the system and negatively impact the network latency. Figure 13 shows that the network latency is affected by the transition latency when the epoch size ranges from 0.5 K to 1 K cycles. A larger epoch size can impact the performance due to the coarse-grain control, and the simulation results show that the epoch size can lead to approximately 4 percent more energy consumption.

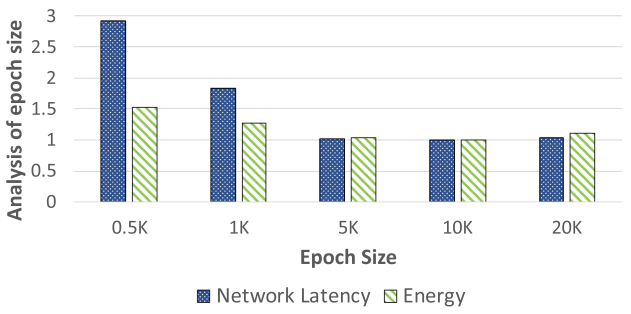

FIGURE 13. Analysis of the epoch size, normalized to 10K cycles.

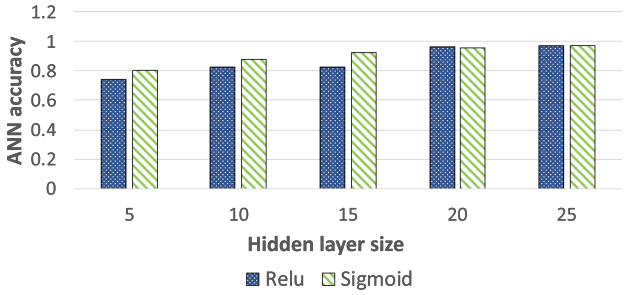

FIGURE 14. Accuracy analysis of the ANN using different sizes of the hidden layer.

B. IMPACTS OF HIDDEN LAYER SIZE ON ANN

The activation function and number of neurons used in the hidden layer of the ANN can affect the accuracy of calculating Q(s,a), thereby determining the network performance. To obtain the optimal performance, we study the ANN accuracy using different types of activation functions and numbers of neurons in the hidden layer. We first explore the use of relu and sigmoid functions. Figure 14 shows that the sigmoid function can achieve higher prediction accuracy when the hidden layer is smaller than 20. When using the sigmoid function, the hidden layer size of 5, 10, 15, 20 and 25 neurons yield accuracies of 80, 87, 90, 96 and 97 percent, respectively, compared with a state-action table. This implies that the accuracy of this network can approach 1 when the hidden layer size equals 20. In this case, a larger number of hidden layer size could barely improve the prediction accuracy but require additional timing and area overhead to calculate and store the weights. Thus, we only apply a single hidden layer to approximate the cost-prohibitive Q-table.

C. IMPACTS OF DISCOUNT FACTOR ON THE RL

To understand the impacts of discount factor on the system performance, we vary the discount factor from 0.1 to 1. Figure 15 shows the results of different discount factors. We observe that a larger discount factor considers the system performance to a greater extent, while a smaller discount factor only considers the power savings. This outcome is due to the fact that the smaller discount factor only considers the immediate reward and excludes the impacts of future rewards. As a result, the near-sighted policy prefers selecting an action with lower power consumption. Compared with a γ of 0.1, a γ of 0.95 yields 13 percent latency reduction but 4 percent energy penalties. On the other hand, compared

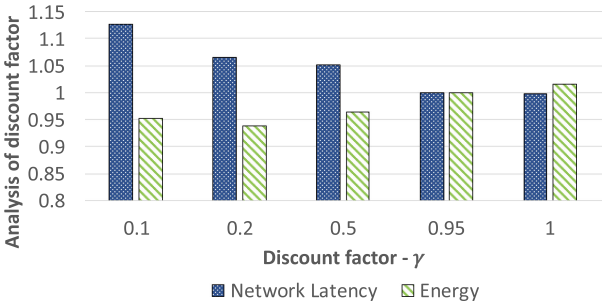

FIGURE 15. **Analysis of the discount factor γ normalized to a γ of 0.95.**

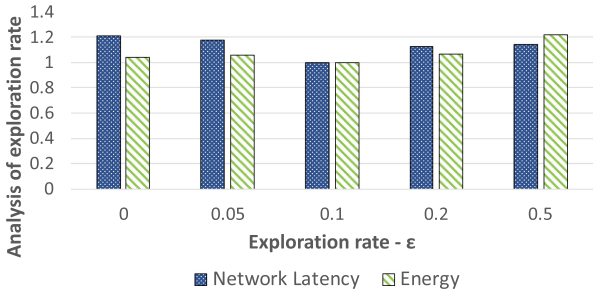

FIGURE 16. **Analysis of the exploration rate ϵ normalized to an ϵ of 0.1.**

with a γ of 1, a γ of 0.95 has similar performance but 3 percent more energy savings.

D. IMPACTS OF EXPLORATION RATE ON RL

To study the impacts of different exploration rates on performance, we vary the exploration rate from 0 to 0.5. Figure 16 shows that the exploration rate of 0.1 has the best network latency performance and energy efficiency. When the exploration rate approaches 0, the policy only uses current policy without any exploration. Since we initialize the policy with reduced voltage/frequency levels, and thus the exploration rate of 0 can lead to 20 percent more latency and 4 percent more energy than the optimal point. For a larger exploration rate, the policy aggressively explores the actions that lead to both performance and energy loss. An exploration rate of 0.5 corresponds to approximately 10 percent latency penalty and 20 percent energy penalty overheads.

VII. RELATED WORKS

A. POWER GATING

In [18], authors proposed to use look-ahead of routing to early wake up the powered-off routers. However, the early wake-up signal cannot fully overlap the wake-up latency which still results in significant network latency. In [19], a fine-grained power-gating scheme has been proposed to dynamically power off different router components. However, the resulting power supplies require prohibitive hardware costs as compared to the coarse-grained power-gating schemes. NoRD [27] implemented a bypass ring network to handle intermittent flits when routers are powered-off. The bypass ring can effectively extend the power-gating benefits and avoid frequent router wake-ups, but it suffers from the scalability issue when NoC size increases. Powerpunch [56] improves power-gating efficiency by sending an early wake-up signal to the powered-off router while the NI is processing the packet. The early notification can wake up the router before the flits arrive. By doing so, the full wake-up latency can be hidden but at the expense of reduced powered-off time. In MP3 [57], the authors used a Clos topology to provide multiple paths for different pairs of source and destination. The intermittent flits can select alternative ways to reach their destinations without waking up powered-off routes. However, the high-radix nature of the Clos network used is cost prohibitive. In [58], authors proposed a multi-network design to maintain the network connectivity. However, the sub-network design results in reduced channel bandwidth, leading to additional serialization latency. Router parking [59] proactively powers off a set of routers and uses an adaptive routing algorithm to detour the packets. However, the detoured packets still requires additional network latencies, affecting application performance. Dim-NoC exploits non-volatile memory technologies to replace power-hungry SRAM-based NoC buffers [60], however, requires significant amount of time overhead for buffer write. Panthre [61] deploys a reconfigurable NoC topology which detours the packets to avoid frequent router wake-ups, trading NoC performance for additional power reduction. SMART [62] uses XY-YX routing and mapping policy to aggregate traffic load to dedicated path between source and destination. However, the XY-YX routing could result in uneven network utilization. SPONGE [63] allows the packets to traverse straightforward without waking up the routers, thereby maximizing static power reduction.

B. DVFS

In [64], DVFS is effectively applied to support on-chip voltage scaling by deploying on-chip voltage regulators. Shang *et al.* [65] exploited DVFS to change the voltage and frequency levels of individual links according to historical link and input buffer utilization. Ogras *et al.* [22] proposed a state feedback approach to predict the workload and achieve better performance-power trade-off. In [24], Chen *et al.* proposed a throughput-driven and a latency-based PI controller for last level caches and NoC, both with dynamic reference points. In [66], the authors used cache-coherence communication properties to predict the NoC traffic and set voltage and frequency. In [23], the authors used shared communication characteristics to categorize the critical information, which results in better power savings. In [54], the authors used a supervised learning algorithm, ridge regression, to predict network utilization. In [25], zhou *et al.* took advantage of the varying timing to merge or separate the router pipeline stages, minimizing the performance overhead caused by reduced router frequency. Similarly, in [67], [68], authors proposed reconfigurable arbitration and allocation logic which adapts to the frequency scaling, maintaining the router performance. Furthermore, significant research [69], [70] reduced frequency of NoC routers

(e.g., buffers, crossbar) to compensate the additional router resources, thereby improving NoC throughput.

C. MACHINE LEARNING IN COMPUTER ARCHITECTURE

In [71], Bai *et al.* proposed to use the RL to select different types of voltage regulator to yield optimal performance and power improvements. Yin *et al.* [72] proposed a self-learned NoC arbitration policy to improve arbitration efficiency. The RL dynamically analyzes various network parameters to automate a policy that reduces network latency. Wang *et al.* [73], [74] used the RL to balance the trade-offs among NoC performance, energy efficiency, and reliability. Specifically, the RL is used to dynamically analyze the dynamic interactions among techniques used for improving NoC fault-tolerant. Won *et al.* [75] proposed an artificial neural network-based DVFS technique that dynamically tunes the voltage and frequency of the last-level caches and NoC. Zheng *et al.* [76] exploited the use of reinforcement learning to balance the static and dynamic power savings, thereby maximizing overall NoC energy savings. In [77], authors proposed to use deep reinforcement learning to control the number and frequency of heterogeneous cores, and thus resulting in optimal power and performance. Li *et al.* [78] exploited the use of supervised learning in router design to identify and predict the variation of traffic pattern in heterogeneous manycore architectures.

VIII. CONCLUSION

In this paper, we proposed Agile, a reinforcement learning (RL)-based NoC design that uniquely combines power-gating and DVFS with the goal of maximizing NoC power savings and improving performance. Several architecture designs and an RL-based control policy are proposed to mitigate the negative effects induced by the combined power-gating and DVFS. The Agile architecture includes (1) a bypass switch that avoids frequently waking up the powered-off router, (2) an optimized bypass pipeline that eliminates unnecessary pipeline stages of the switch to reduce network latency, and (3) the reversible links that can be dynamically allocated to where they are needed to improve network throughput. In addition, we use RL to automate a control policy that can select the voltage/frequency level, power-gating decisions, and optimal architecture configurations to provide optimal power savings and performance. Moreover, we explore the use of an artificial neural network (ANN) to efficiently implement RL with minimum storage spaces. Our simulation results show that agile can achieve up to 58 percent overall power savings and 11 percent execution reduction as compared to state-of-art designs. The ANN-based RL and bypass switch incur nominal area overhead of 5 percent as compared to a conventional router.

ACKNOWLEDGMENTS

This research was partially supported by NSF Grants CCF-1420718, CCF1513606, CCF-1703013, CCF-1547034, CCF-1547035, CCF-1540736, and CCF-1702980. The authors sincerely thank the anonymous reviewers for their excellent feedback..

REFERENCES

[1] L. Benini and G. De Micheli , "Networks on chips: A new SoC paradigm," *IEEE Comput.*, vol. 35, no. 1, pp. 70–78, Jan. 2002.
[2] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam, The Netherlands: Elsevier, 2004.
[3] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks*. San Mateo, CA, USA: Morgan Kaufmann, 2003.
[4] Y. Hoskote, S. Vangala, A. Singha, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep./Oct. 2007.
[5] T. Mattson *et al.*,"The 48-core SCC processor: The programmer's view," in *Proc. ACM/IEEE Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2010, pp. 1–11.
[6] V. Ganesh *et al.*, "Conservation cores: Reducing the energy of mature computations," in *Proc. ACM Int. Conf. Architect. Support Program. Languages Operating Syst.*, 2010, pp. 205–218.
[7] H. Alkhatib *et al.*"What will 2022 look like? the IEEE CS 2022 report," *IEEE Comput.*, vol. 48, no. 3, pp. 68–76, Mar. 2015.
[8] S. Bell *et al.*, "Tile64-processor: A 64-core SoC with mesh interconnect," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2008, pp. 88–598.
[9] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proc. Int. Symp. Comput. Architecture*, 2009, pp. 196–207.
[10] F. Alazemi, A. Azizimazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2018, pp. 492–503.
[11] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," in *Proc. Int. Symp. Comput. Architecture*, 2007, pp. 150–161.
[12] A. Kodi, A. Sarathy, and A. Louri, "ideal: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures," in *Proc. IEEE/ACM Int. Symp. Comput. Architecture*, 2008, pp. 241–250.
[13] H. Wang, L. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2003, Art. no. 105.
[14] 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 *Proc. ACM/IEEE Des. Autom. Conf.*, 2014, pp. 1–6.
[15] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2011, pp. 144–155.
[16] Y. Chen, M. F. Reza, and A. Louri, "DEC-NoC: An approximate framework based on dynamic error control with applications to energy-efficient NoCs," in *Proc. IEEE 36th Int. Conf. Comput. Des.*, 2018, pp. 480–487.
[17] H. Zheng, K. Wang, and A. Louri, "A versatile and flexible chiplet-based system design for heterogeneous manycore architectures," in *Proc. ACM/IEEE 57th Annu. Des. Automat. Conf.*, 2020, pp. 1–6.
[18] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Run-time power gating of on-chip routers using look-ahead routing," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2008, pp. 55–60.
[19] 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 *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, 2010, pp. 61–68.
[20] M. Hiroki, K. Michihiro, and A. Hideharu, "Run-time power gating of on-chip routers using look-ahead routing," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2008, pp. 55–60.
[21] A. Mirhosseini, M. Sadrosadati, A. Fakhrzadehgan, M. Modarressi, and H. Sarbazi-Azad, "An energy-efficient virtual channel power-gating mechanism for on-chip networks," in *Proc. Des. Autom. Test Eur. Conf.*, 2015, pp. 1527–1532.
[22] U. Ogras, R. Marculescu, D. Marculescu, and E. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 330–341, Mar. 2009.
[23] Y. Yao and Z. Lu, "DVFS for NoCs in CPPs: A thread voting approach," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2016, pp. 309–320.

[24] X. Chen *et al.*, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *Proc. ACM/IEEE Annu. Des. Autom. Conf.*, 2013, pp. 1–7.

[25] P. Zhou, J. Yin, A. Zhai, and S. Sapatnekar, "NoC frequency scaling with flexible-pipeline routers," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Des.*, 2011, pp. 403–408.

[26] A. Mishra, R. Das, S. Eachempati, R. Iyer, and N. Vijaykrishnan, and C. R. Das, "A case for dynamic frequency tuning in on-chip networks," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 292–303.

[27] L. Chen and T. Pinkston, "NoRD: Node-router decoupling for effective power-gating of on-chip routers," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2012, pp. 270–281.

[28] H. Zheng and A. Louri, "EZ-pass: An energy & performance-efficient power-gating router architecture for scalable NoCs," *IEEE Comput. Architecture Lett.*, vol. 17, no. 1, pp. 88–91, Jan./Jun. 2018.

[29] M. Badr and N. Jerger, "Synfull: Synthetic traffic models capturing cache coherent behaviour," in *Proc. Int. Symp. Comput. Architecture*, 2014, pp. 109–120.

[30] Y. Yao and Z. Lu, "iNPG: Accelerating critical section access with in-network packet generation for NoC based many-cores," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2018, pp. 15–26.

[31] T. Jain, M. Ramakrishna, P. Gratz, A. Sprintson, and G. Choi, "Asynchronous bypass channels for multi-synchronous NoCs: A router microarchitecture, topology, and routing algorithm," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 11, pp. 1663–1676, Nov. 2011.

[32] L. Xin and C. Choy, "A low-latency NoC router with lookahead bypass," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 3981–3984.

[33] L. Peh and W. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. IEEE Int. Symp. High-Perform. Comput. Architecture*, 2001, pp. 255–266.

[34] D. Becker and W. Dally, "Allocator implementations for network-on-chip routers," in *Proc. IEEE Conf. High Perform. Comput. Netw. Storage Anal.*, 2009, pp. 1–12.

[35] O. Chen, S. Park, T. Krishna, S. Subramanian, A. Chandrakasan, and L. Peh, "Smart: A single-cycle reconfigurable NoC for SoC applications," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2013, pp. 338–343.

[36] H. Farrokhbakht, M. Taram, B. Khaleghi, and S. Hessabi, "Toot: An efficient and scalable power-gating method for NoC routers," in *Proc. IEEE/ACM Int. Symp. Netw.-on-Chip*, 2016, pp. 1–8.

[37] R. Boyapati, J. Huang, N. Wang, K. Kim, K. Yum, and E. Kim, "Fly-over: A light-weight distributed power-gating mechanism for energy-efficient networks-on-chip," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2017, pp. 708–717.

[38] D. DiTomaso, A. Kodi, and A. Louri, "QORE: A fault tolerant network-on-chip architecture with power-efficient quad-function channel (QFC) buffers," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2014, pp. 320–331.

[39] R. Hesse, J. Nicholls, and N. Jerger, "Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels," in *Proc. IEEE/ACM Int. Symp. Netw.-on-Chip*, 2012, pp. 132–141.

[40] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. Yousif, and C. Das, "Vichar: A dynamic virtual channel regulator for network-on-chip routers," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2006, pp. 333–346.

[41] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[42] L. Busoniu *et al.*, "Multi-agent reinforcement learning: A survey," in *Proc. Int. Conf. Control Autom. Robot. Vis.*, 2006, pp. 1–6.

[43] A. Mishra, O. Mutlu, and C. Das, "A heterogeneous multiple network-on-chip design: An application-aware approach," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2013, pp. 1–10.

[44] R. Das *et al.*, "Application-aware prioritization mechanisms for on-chip networks," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 280–291.

[45] A. Bakhoda, J. Kim, and T. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2010, pp. 421–432.

[46] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.

[47] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[48] N. Agarwal, T. Krishna, L. Peh, and N. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2009, pp. 33–42.

[49] C. Sun *et al.*, "DSENT a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Proc. ACM/IEEE Int. Symp. Netw. Chip*, 2012, pp. 201–210.

[50] J. Stine *et al.*, "Freepdk: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectronic Syst. Educ.*, 2007, pp. 173–174.

[51] C. Bienia, S. Kumar, J. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Architectures Compilation Techn.*, 2008, pp. 72–81.

[52] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[53] X. Chen, Z. Xu, H. Kim, P. Gratz, J. Hu, M. Kishinevsky, and U. Ogras, "In-network monitoring and control policy for DVFS of CMP networks-on-chip and last level caches," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 4, 2013, Art. no. 47.

[54] M. Clark, R. Bunescu, A. Kodi, and A. Louri, "Lead: Learning-enabled energy-aware dynamic voltage/frequency scaling in NoCs," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2018, pp. 1–6.

[55] T. Krishna, C.-H. O. Chen, and W. C. Kwon, and L.-S. Peh, "Breaking the on-chip latency barrier using smart," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2013, pp. 378–389.

[56] L. Chen, D. Zhu, M. Pedram, and T. Pinkston, "Power punch: Towards non-blocking power-gating of NoC routers," in *Proc. IEEE Int. Symp. High Performance Comput. Architecture*, 2015, pp. 378–389.

[57] L. Chen, L. Zhao, and T. M. Pinkston, "MP3: Minimizing performance penalty for power-gating of clos network-on-chip," in *Proc. IEEE Int. Symp. High-Perform. Comput. Architecture*, 2014, pp. 296–307.

[58] R. Das, S. Narayanasamy, S. Satpathy, and R. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," in *Proc. IEEE/ACM Int. Symp. Comput. Architecture*, 2013, pp. 320–331.

[59] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *Proc. 19th IEEE Int. Symp. High Perform. Comput. Architecture*, 2013, pp. 508–519.

[60] Z. Jia, O. Jin, G. Fen, Z. Jishen, and X. Yuan, "DimNoC: A dim silicon approach towards power-efficient on-chip network," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2015, pp. 1–6.

[61] P. Ritesh, D. Reetuparna, and B. Valeria, "Power-aware NoCs through routing and topology reconfiguration," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2014, pp. 1–6.

[62] H. Farrokhbakht, H. Kamali, and S. Hessabi, "Smart: A scalable mapping and routing technique for power-gating in NoC routers," in *Proc. IEEE/ACM Int. Symp. Netw.-on-Chip*, 2017, Art. no. 15.

[63] H. Farrokhbakht, H. Kamali, N. Jerger, and S. Hessabi, "Sponge: A scalable pivot-based on/off gating engine for reducing static power in NoC routers," in *Proc. Int. Symp. Low Power Electron. Des.*, 2018, Art. no. 17.

[64] W. Kim, M. Gupta, G. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2008, pp. 123–134.

[65] L. Shang, L. Peh, and N. Jha, "Power-efficient interconnection networks: Dynamic voltage scaling with links," *IEEE Comput. Architecture Lett.*, vol. 1, no. 1, pp. 6–6, Jan. 2002.

[66] R. Hesse and N. Jerger, "Improving DVFS in NoCs with coherence prediction," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, 2015, Art. no. 24.

[67] M. Sadrosadati, A. Mirhosseini, H. Aghilinasab, and H. Sarbazi-Azad, "An efficient DVS scheme for on-chip networks using reconfigurable virtual channel allocators," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Des.*, 2015, pp. 249–254.

[68] A. Mirhosseini, M. Sadrosadati, F. Aghamohammadi, M. Modarressi, and H. Sarbazi-Azad , "BARAN: Bimodal adaptive reconfigurable-allocator network-on-chip," *ACM Trans. Parallel Comput.*, vol. 5, no. 3, 2019, Art. no. 11.

[69] H. Matsutani, M. Koibuchi, and D. Wang, and H. Amano, "Adding slow-silent virtual channels for low-power on-chip networks," in *Proc. ACM/IEEE Int. Symp. Netwo.-on-Chip*, 2008, pp. 23–32.

[70] A. Mirhosseini, M. Sadrosadati, B. Soltani, H. Sarbazi-Azad, and T. Wenisch, "Binochs: Bimodal network-on-chip for CPU-GPU heterogeneous systems," in *Proc. IEEE/ACM Int. Symp. Netw.-on-Chip*, 2017, Art. no. 7.

[71] Y. Bai, V. Lee, and E. Ipek, "Voltage regulator efficiency aware power management," in *Proc. ACM Int. Conf. Architect. Support Program. Languages Operating Syst.*, 2017, pp. 825–838.

[72] J. Yin, Y. Eckert, S. Che, M. Oskin, and G. Loh, "Toward more efficient NoC arbitration: A deep reinforcement learning approach," in *Proc. 1st Int. Workshop AI-Assisted Des. Architecture*, 2018, pp. 1–6.

[73] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "High-performance, energy-efficient, fault-tolerant network-on-chip design using reinforcement learning," in *Proc. 22nd Des. Autom. Test Eur. Conf.*, 2019, pp. 1166–1171.

[74] 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. IEEE/ACM Int. Symp. Comput. Architecture*, 2019, pp. 589–600.

[75] 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. IEEE Int. Symp. High Perform. Comput. Architecture*, 2014, pp. 308–319.

[76] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2019, Art. no. 47.

[77] U. Gupta, S. Mandal, M. Mao, C. Chakrabarti, and U. Ogras, "A deep Q-learning approach for dynamic management of heterogeneous processors," *IEEE Comput. Architecture Lett.*, vol. 18, no. 1, pp. 14–17, Jan.-Jun. 2019.

[78] Y. Li and A. Louri, "ALPHA: A learning-enabled high-performance network-on-chip router design for heterogeneous manycore architectures," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2020.2981340.

HAO ZHENG (Student Member, IEEE) received the BS degree in electrical engineering from Beijing Jiao Tong University, Beijing, China, and the MS degree in electrical engineering from George Washington University, Washington, DC, where he is currently working toward the PhD degree in computer engineering. His research interests include the areas of computer architecture and parallel computing, with emphasis on on-chip interconnects and energy-efficient manycore architecture designs.

AHMED LOURI (Fellow, IEEE) received the PhD degree in computer engineering from the University of Southern California, Los Angeles, California, in 1988. He is the David and Marilyn Karlgaard Endowed chair professor of Electrical and Computer Engineering with the George Washington University, and the director of the High Performance Computing Architectures and Technologies Laboratory. From 2010 to 2013, he served as a program director with the National Science Foundations (NSF) Directorate for Computer and Information Science and Engineering. He conducts research with the broad area of computer architecture and parallel computing, with emphasis on interconnection networks, optical interconnects for scalable parallel computing systems, reconfigurable computing systems, and power-efficient and reliable Network-on-Chips (NoCs) for multicore architectures. He is currently serving as the editor-in-chief of the *IEEE Transactions on Computers*.