CURE: A High-Performance, Low-Power, and Reliable Network-on-Chip Design Using Reinforcement Learning

Ke Wang, *Student Member, IEEE* and Ahmed Louri, *Fellow, IEEE*

Abstract—We propose CURE, a deep reinforcement learning (DRL)-based NoC design framework that simultaneously reduces network latency, improves energy-efficiency, and tolerates transient errors and permanent faults. CURE has several architectural innovations and a DRL-based hardware controller to manage design complexity and optimize trade-offs. First, in CURE, we propose reversible multi-function adaptive channels (RMCs) to reduce NoC power consumption and network latency. Second, we implement a new fault-secure adaptive error correction hardware in each router to enhance reliability for both transient errors and permanent faults. Third, we propose a router power-gating and bypass design that powers off NoC components to reduce power and extend chip lifespan. Further, for the complex dynamic interactions of these techniques, we propose using DRL to train a proactive control policy to provide improved fault-tolerance, reduced power consumption, and improved performance. Simulation using the PARSEC benchmark shows that CURE reduces end-to-end packet latency by 39 percent, improves energy efficiency by 92 percent, and lowers static and dynamic power consumption by 24 and 38 percent, respectively, over conventional solutions. Using mean-time-to-failure, we show that CURE is 7.7× more reliable than the conventional NoC design.

Index Terms—Computer architecture, network-on-chip(NoC), reliability, deep reinforcement learning

1 Introduction

Network-on-Chips (NoCs) [1], [2] have emerged as the standard interconnect solutions for connecting multiple cores, memory modules, and other hardware components. With continuous aggressive technology scaling, the reliability issue of NoCs becomes considerably more pronounced because the transistors and wires in NoCs are becoming increasingly vulnerable to faults, which are predominantly classified as permanent faults (induced by hardware aging) and transient errors (caused by runtime variations, overheated hardware, transistor delays, etc.).

A significant amount of work has been proposed to enhance the robustness of NoC [3], [4], [5], [6], [7], [8], [9], [10]. Unfortunately, these existing fault-tolerant methodologies have some critical defects. First, these techniques are limited, because they only focus on either faults within routers (at the gate-level) [3], [4] or faults that occur on inter-router links (at the link-level) [6], [7], [8], [9], [10]. Second, these conventional error-handling techniques can be expensive: SHIELD [3] and Vicis [4] duplicate the logic circuitry in routing pipeline stages or use redundant input ports, which consume massive chip area, while other retransmission-based fault-handling schemes, such as [5] and [6], generally incur substantial power consumption and prohibitive latency.

Deploying power-saving and performance-enhancing techniques to compensate the cost and performance degradation caused by fault-tolerant methodologies is indispensable yet extremely complex. Different optimization techniques, when being used simultaneously, can conflict and offset each other's desired goals, which presents various design trade-offs. For example, channel buffers [7], [11], [12] replace the power-consuming router buffers with link storage to save power, but result in performance loss due to the limited throughput of link storage. Power-gating techniques [13], [14], [15] are proposed to take advantage of idle router periods to save power; however, they incur prohibitive wake-up latency. Dynamic voltage and frequency scaling [13], [16] intends to balance power savings and network throughput, but it can lead to increased transient faults [17]. Due to the explosion and complexity of the design space, we intend to use machine learning techniques to optimize the dynamic interactions of different techniques and automatically learn an optimal control policy to address the challenges of simultaneously decreasing power consumption, increasing performance, and improving reliability.

In this paper, we propose *CURE*, a learning-based NoC design framework, that handles permanent and transient faults at both the gate-level and link-level in a high-performance, low-power-consumption manner. CURE uniquely utilizes a per-router deep reinforcement learning (DRL)-based [18], [19], [20] control policy to explore the dynamic interactions among NoC components and system-level performance metrics as well as optimizing the trade-offs

• *The authors are with the Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052. E-mail: {cory, louri}@gwu.edu.*

Manuscript received 16 July 2019; revised 18 Dec. 2019; accepted 4 Apr. 2020. Date of publication 8 Apr. 2020; date of current version 1 May 2020. (Corresponding authors: Ke Wang.) Recommended for acceptance by M. Guo. Digital Object Identifier no. 10.1109/TPDS.2020.2986297

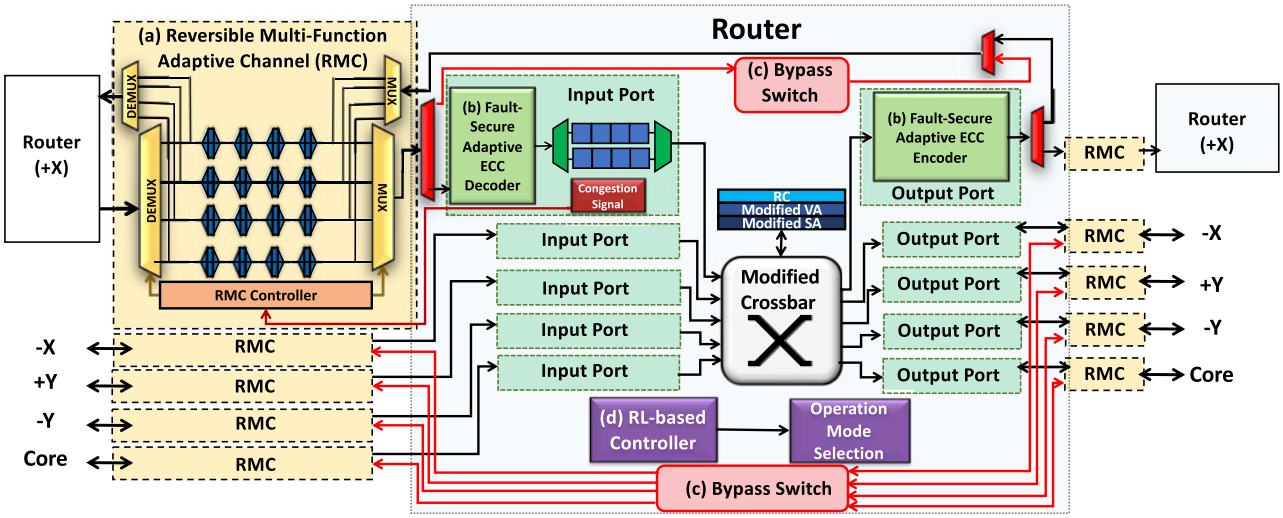

Fig. 1. CURE architecture design. CURE consists of (a) a reversible multi-function adaptive channel (RMC) between adjacent routers, (b) a new fault-tolerant router design with modified VA & SA & ST (crossbar) and adaptive error detection/correction hardware, (c) a router bypass route, and (d) a reinforcement learning (DRL)-based control policy.

at runtime. The major contributions of this paper are as follows:

- *Improved Inter-Router Channel Design:* We improve our previous proposed channel buffers [8] by designing reversible multi-function adaptive channel (RMC) buffers. RMC buffers have three basic functions: (1) forward/backward regular repeaters for flit propagation, (2) forward/backward buffers for link storage, and (3) forward/backward re-transmission buffers for handling faults. RMC is beneficial in several ways. First, with the additional storage available on the inter-router channel, dynamic power consumption for on-chip storage is reduced at high network loads without any performance degradation. Second, RMC provides flexibility for improving reliability at the link-level via re-transmission buffers and reversing the propagation direction to avoid faulty links. Third, the reversibility of RMC enhances performance by allowing the NoC to dynamically adapt to traffic because it provides extra link bandwidth in a specific direction at high network loads. Additionally, we utilize the bypass route proposed in [8], for the purpose of enhancing reliability, to retain NoC connectivity when permanent faults occur on RMCs.
- *Robust Router Microarchitecture Design:* We significantly improve the NoC robustness for both transient errors and permanent faults. First, we propose per-router self-diagnosis adaptive error control hardware to detect/correct faults at the link-level. The proposed error correction hardware adapts to the error level of each port and dynamically deploys the most efficient error detection/correction and flit re-transmission schemes with minimized power and latency overheads. Additionally, a low-cost, fault-vulnerable detector is implemented inside the error control hardware to detect malfunctions of the error control hardware itself. Second, we modify the circuitry of the router to mitigate transient errors and permanent faults that occur in the routing pipeline stages at the gate-level. Additionally, we propose a power-gating scheme that dynamically powers off NoC components (router buffers, crossbar, error control hardware, etc.) when needed to achieve power savings and mitigate aging effects.
- *DRL-Based Control Policy Design:* We propose a set of unique operation modes for each router with a DRL-based control policy to handle the dynamic interactions and optimize the trade-offs. The goal of dynamically utilizing different operation modes is to achieve improved performance, maximized power savings, and enhanced reliability. At runtime, per-router DRL agents observe and learn from the entire NoC environment and automatically evolve optimal per-router control policies that select the optimal operation modes at any given time.

We evaluate the performance of the proposed CURE architecture using a modified Booksim2 [21] simulator with PARSEC benchmarks on an 8×8 2D mesh architecture. We show that the proposed *CURE* provides significant power savings, enhanced reliability, higher performance, and lower area overhead compared to multiple state-of-the-art NoC designs with power-saving and fault-tolerant mechanisms.

2 CURE Microarchitecture

In this section, we demonstrate the architectural innovations of the proposed framework. The overall microarchitecture of the proposed design is shown in Fig. 1. The proposed design consists of inter-router links based on reversible multi-function channels (RMCs), a fault-tolerant router design, dynamic fault-secure error correction hardware, and a router bypass route for power savings and stress management. Additionally, a DRL-based controller is located in each router to handle the dynamic interactions and optimize the trade-offs. The implementation of RMCs is described in Section 2.1. The fault-tolerant router design is presented in Section 2.2. The adaptive error correction (ECC) hardware is presented in Section 2.3, and the bypass route is discussed in Section 2.4.

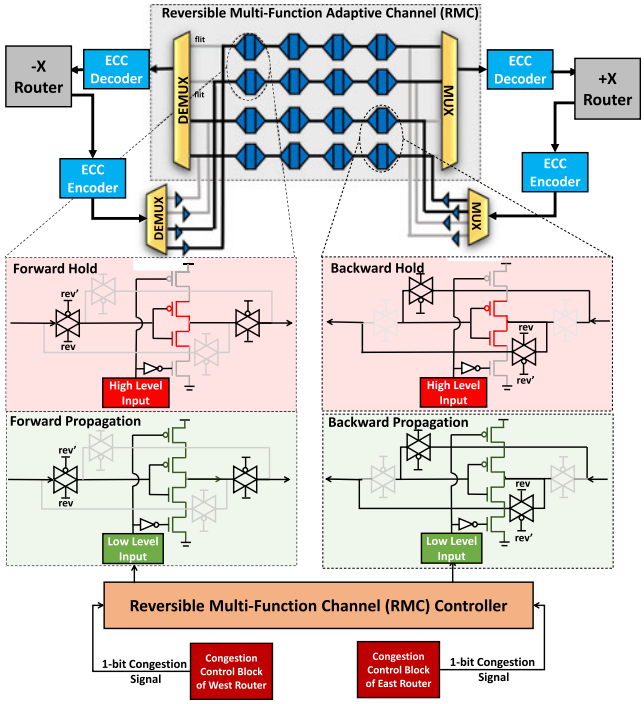

Fig. 2. Proposed reversible multi-function adaptive channel (RMC). Each RMC consists of four physical links with four buffer stages per link.

2.1 Reversible Multi-Function Channels (RMCs)

Although previous research [7], [11], [12] has shown that the excessive power consumption of router buffers can be reduced by moving storage to inter-router links or channels, the performance loss has not been thoroughly considered. In this paper, we borrow the idea of reversible links in [7] and extend the multi-function adaptive channel (MFAC) buffers in [8] to reversible multi-function adaptive channel buffers (RMC buffers), with the objective of improving network performance. With the newly designed channel architecture, RMC storage, links, and router storage are dynamically allocated according to traffic patterns to reduce power consumption. As shown in Fig. 2, each RMC buffer consists of an inverter and two tri-state transistors to enable store/propagation and four transmission control gates to reverse the link direction. The reversibility of RMC allows the links to adapt to different traffic loads for enhanced network throughput and the wear-out on different links for enhanced reliability. Each RMC between two adjacent routers has four physical links, and the channel buffers are evenly allocated on those links.

Next, we use an RMC controller to dynamically and independently configure the transmission/buffer functions of the physical links to perform multiple RMC functions, as shown in Fig. 3. As described in detail in Section 2.1.1, RMCs can function as (1) forward/backward transmission repeaters, (2) forward/backward link storage, and (3) forward/backward re-transmission buffers. The dynamic selection of RMC functionalities is explained in Section 3. Because our design may lead to a latency penalty (due to the control overhead of the RMC buffers) and potential congestion (due to the head-of-line blocking of router buffers), we use dynamic router buffer allocation to maximize network throughput, as detailed in Section 2.1.2.

2.1.1 RMC Buffer Functionalities

Previous designs [8], [12] show that tri-state transistors can be used for propagating or storing flits. Fig. 2 shows the proposed RMC buffers based on those tri-state transistors. The proposed RMC between two adjacent routers consists of four physical links, with four buffer stages per link. Each physical link can be used for either storage or transmission as regular repeaters, in both directions, controlled by the additional four transmission gates. The direction and functionality of each RMC link are configured by the added RMC controller. The RMC controller first uses the a single-bit reversal (*rev* and *rev'*) signal to enable/disable the four transmission gates. In this way, the link can be configured in a specific direction. For example, when the rev signal is high, the RMC buffer will store/propagate flits forward. After configuring the direction, the RMC controller will send a function selection signal along with the 1-bit congestion signal from the downstream router to enable one of the functions of the RMC buffer. The RMC buffer can implement three basic functions in both directions, namely, link storage, transmission repeaters, and re-transmission buffers, which are illustrated in Fig. 3 and discussed below.

(1) Forward/Backward Transmission Repeater (Fig. 3a): In this case, the RMC buffer links are configured as repeaters. When the RMC controller is set to forward the congestion signal and the 1-bit congestion signal is low, the transistors connected to GND and V_{dd} are enabled, allowing the RMC to act as a transmission channel.

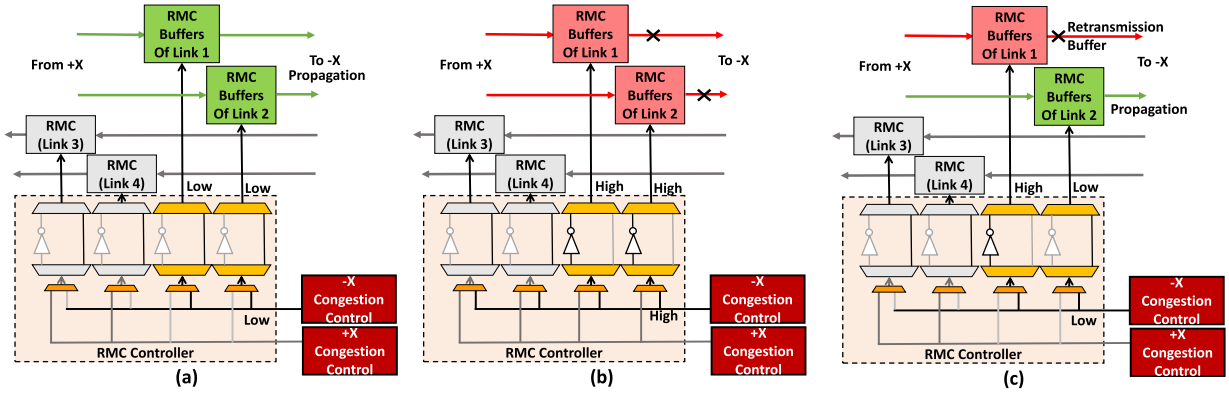

Fig. 3. Multi-function adaptive channel (RMC) buffers assume three different functions: (a) forward/backward regular repeaters for flit transmission, (b) forward/backward link buffers for storage on the link itself, and (c) forward/backward re-transmission buffers to store a copy of error-free flit for fault-tolerance.

(2) Forward/Backward Link Storage (Fig. 3b): In this case, the RMC buffer links are configured as link storage. When the RMC controller is set to forward the congestion signal and the 1-bit congestion signal is high, the transistors connected to GND and V_{dd} are disabled. Flits are then buffered in the transistors' capacitance.

(3) Forward/Backward Re-transmission Buffer (Fig. 3c): This functionality can only be activated when at least two physical RMC links are in the same direction. In this case, we use one of the RMC buffer links to store flits for re-transmission purposes, whereas all other RMC buffer links are used for transmission. In conventional re-transmission-based error control design, a copy of the transmitted flit is stored in the local re-transmission buffer (in the upstream router) until it receives an acknowledgment (ACK) message back from the downstream router. The implementation of local re-transmission buffers can lead to excessive power and area overhead, especially when these re-transmission buffers are underutilized in low-error scenarios. Therefore, replacing the traditional in-router re-transmission buffers with RMC buffers is beneficial because the original flit will only be stored when needed (under higher error rates). Under this condition, the RMC controller will send the same packets/flits to both RMC buffer links. The RMC controller configures one of the RMC buffer links for storage (by applying a "hold" signal) and the other RMC buffer links for forwarding the flit (regular transmission). Upon receiving a NACK signal, the RMC controller releases the flit for re-transmission. If an ACK signal is received, the flit is discarded because the original transmission is error-free.

2.1.2 RMC Buffer Allocation and Flow Control

In CURE, to ensure connectivity in all directions, at least one of the four physical links in each RMC is allocated to each direction. For instance, as shown in Fig. 2, the top link is always facing -X, while the bottom link is facing +X. However, the middle links can be dynamically configured to face either direction. Therefore, the RMC has three different configurations (with the top link always facing -X, and the bottom link always facing +X):(1) the middle links are facing -X and +X, (2) both the middle links are facing -X, and (3) both the middle links are facing +X. For the ease of explanation, we name these three configurations $RMC_{2:2}$, $RMC_{3:1}$, and $RMC_{1:3}$, according to the link number of each direction.

As mentioned above, determining which direction to allocate the RMC links is critical. Network traffic is measured using link utilization and buffer utilization values monitored by the corresponding router. The RMC controller analyzes the network throughput in each direction, using the number of propagated packets, and calculates the ratio of these two throughput numbers. The RMC controller will allocate the RMC links with the configuration that is nearest to the ratio. For example, if the ratio of the -X traversal to the +X traversal is 2.6, the $RMC_{3:1}$ configuration will be utilized.

After allocating the direction of RMC links, the RMC buffers will be allocated using a unified buffer state table (BST), as shown in Fig. 4. The proposed BST is router-associated and shared by all the input ports within the router and remains accessible even if the router is power-gated (power-gating information is recorded in the yellow and orange

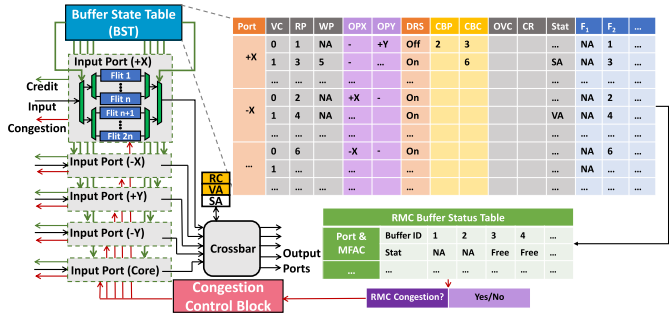

Fig. 4. Proposed unified buffer state table (BST). The green arrows indicate buffer slot allocation and credit signals by BST, while congestion signals are shown with red arrows.

entries shown in Fig. 4). The proposed BST is a modified version of the conventional virtual channel (VC) state table. The conventional VC table consists of the following information, or entries: the VC identifier (VC), read pointer (RP), write pointer (WP), allocated output port (OP), output VC (OVC), status (Stat), and credit count (CR). In the conventional VC state table, the header flit carries the packet information for route computation (RC) and VC allocation. The VC state table allocates a free VC slot to the header flit and records the VC information (VCID) and output information (output VC and output port). The body flits of the packet simply follow the VCID to find the correct output port from the VC state table. Thus, the packet is routed correctly.

In CURE, we modify the VC state table and implement BST to support RMC channel buffers and record routing information when the router is power-gated. Compared to input-port-associated VC state tables, the proposed BST is router-associated and shared by all input ports within the router. Other than the entries of the conventional VC state table, the proposed BST also consists of additional entries for allocating channel buffers: input port identifier (Port) that indicates the input port of the incoming flit, downstream router status (DRS) that indicates if the downstream router is power-gated, a channel buffer pointer (CBP) and channel buffer credit (CBC) that indicate the occupancy status of the associated RMC buffers. We also create two entries, OPX and OPY, to replace the conventional output port entry to support adaptive XY/YX routing. To enable BST functioning when the router is power-gated, we consider a separate supply voltage that is not powered-off for BST.

The flow control is simple. While the router is powered on, the body flits simply follow the VC and output port information carried by the header flit using the BST. Similarly, when the router is power-gated, the BST also records the VC and OP information of the header flit. Thus, the body flits can be routed to the associated output port by looking up the information. When the flit leaves the bypass switch, a credit is sent back to its upstream router for updating the credit information. This guarantees that the flow control operates normally, irrespective of whether the router is powered on or off. The BST is powered by a separate voltage supply. Additionally, the congestion control block monitors and updates the BST by recording all the available router buffer and RMC buffer slots. If all the router buffer slots and RMC buffer slots of an input direction are occupied, a congestion signal will be triggered.

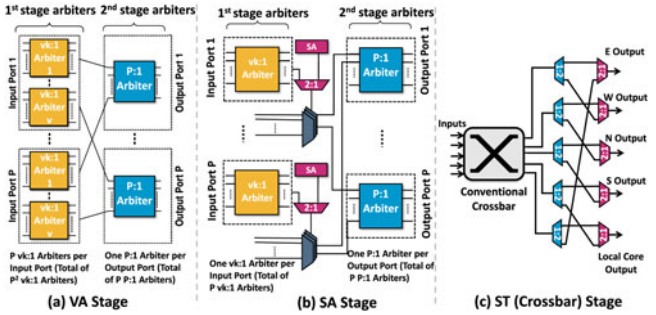

Fig. 5. Proposed fault-tolerant router.

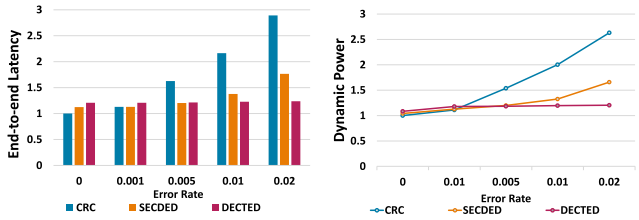

Fig. 6. Trade-offs of different static error control schemes.

To further balance traffic and utilize RMC resources, we propose an adaptive XY/YX routing algorithm to mitigate traffic in high-intensity ports/links to underutilized ports/links after RMC is configured. This can provide more routing options when flits compete for the same output port. At the RC stage, for a packet from the local injection port, the RC unit calculates the output ports out of both the XY and YX routes and stores the results in the corresponding VC state table entry. For a packet from other input ports, the routing identification bit in its head flit decides which routing computation unit is used. Because XY and YX routes can be calculated simultaneously with symmetric logic, the proposed RC unit does not introduce extra delays in the RC stage. The VC allocation stage utilizes the unified BST. All VC states are stored in the unified BST. The OP entry is extended to two parts: the output port for XY routing (OPX) and the output port for YX routing (OPY). For a packet newly injected from the injection port of the current router, the information of the output ports of both XY and YX routing is useful in the VA stage. For a packet just passing by the current router, the routing algorithm has already been determined; thus, only one of two output ports is valid. The other output port will be ignored based on the additional routing identification bit in the head flit. The proposed routing scheme is deadlock-free.

Note that in the 4-pipeline-stage router, the arbitration logic for VA/SA stages dominates the critical path and determines the minimum clock period. Using the Synopsys Design Compiler, we determined the critical delay of VA/SA to be 0.318 nsec (VA)/0.386 nsec (SA) for conventional routers and 0.246 nsec (VA)/0.453 nsec (SA) for CURE. Both of the critical delays are fulfilled within 0.5nsec clock (i.e., 2.0 GHz clock frequency).

2.2 Fault-Tolerant Router Design

A typical NoC router consists of input/output ports, buffers, routing logic, and a crossbar that connects the input ports to output ports for packet routing. The conventional router has four pipeline stages: routing computation stage, virtual channel allocation stage (VA), switch allocation stage (SA), and switch traversing (ST) , which is also known as the crossbar stage. Permanent faults occasionally occur in the last three stages if the corresponding hardware (i.e., arbiter, switch, link, and crossbar) is faulty [3]. In this paper, we propose a new router design that can tolerate permanent faults in the VA, SA, and ST (crossbar) pipeline stages.

VA and SA Stages. The CURE architecture uses the unified BST to allocate VC and switches. Conventional VA has two stages. In the first stage, each input VC that has a head flit arbitrates for an empty VC at the downstream router using the RC results. In the second stage, head flits across different input VCs that have been allocated the same virtual channel in the downstream router compete with each other. Similar to the VA stage, the SA stage also has two stages: the first stage decides which VC of an input port can propagate its flit to the crossbar stage, while the second stage resolves the competition between VCs of different input ports trying to access the same output port. Each of these stages is composed of a set of arbiters associated with a specific VC. Permanent faults may occur when any of the arbiters are faulty.

In this paper, several simple MUXes/DEMUXes are added to the conventional VA and SA stages to allow the router to bypass the faulty arbiter and borrow the unoccupied arbiter to perform VC and switch allocation. We modified the SHIELD [3] architecture to fit the proposed CURE using RMC buffers and unified BST. The proposed architecture is shown in Fig. 5. As shown in Fig. 5, in the VA and SA stages, arbiters from the virtual channels of the same input port can be shared.

ST (Crossbar) Stage. A crossbar is considered faulty if any of the MUXes/DEMUXes located in the crossbar are faulty and the packet fails to be forwarded to the output port. To overcome such faults, additional MUXes and DEMUXes are added to each output port to create a backup flit ejection path. The modified crossbar is shown in Fig. 5c. The crossbar is fault-free as long as either the original path or the backup path is not faulty.

2.3 Fault-Secure Adaptive ECC Hardware Design

Conventional static error correction hardware is either not power efficient or not powerful enough to handle transient faults: lightweight error control schemes (e.g., end-to-end CRC) can lead to excessive re-transmission traffic at high error rates, and powerful error correction schemes (e.g., double-error correction triple-error detection (DECTED)) are power consuming. We use three existing static error control schemes, namely end-to-end CRC, per-hop SECDED, and per-hop DECTED, to evaluate the trade-offs between NoC performance metrics of different static error mitigation techniques. The evaluation result is shown in Fig. 6. As shown in Fig. 6, for low error levels, SECDED and DECTED both negatively impact system performance due to their encoding and decoding overheads. However, as the error level increases, it is beneficial to deploy SECDED and DECTED to mitigate re-transmission packets. Therefore, there is a strong need for a dynamic error control hardware that can adapt to different error levels and apply the most efficient error mitigation technique at runtime. To this end, we propose a per-router-based adaptive error control hardware to

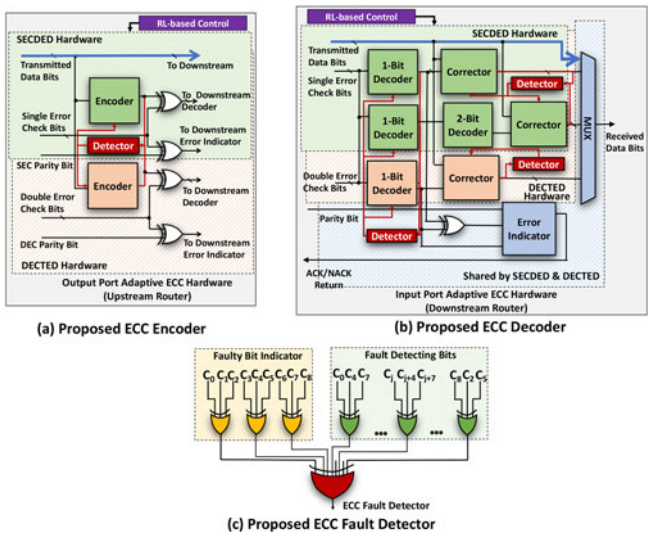

Fig. 7. Proposed fault-secure adaptive error control hardware. (a) Fault-secure adaptive error control hardware, including encoders located in the router's output port and decoders located in the router's input port. SECDED is active when logic circuits in green and blue are enabled, DECTED is active when logic circuits in green, orange, and blue are enabled. The red arrow shows flits with CRC enabled. (b)Proposed ECC fault detector. The green OR gates detect malfunctions in ECC hardware, while the yellow OR gates indicate if a permanent fault occurs.

mitigate soft errors in RMC links for balanced NoC power, performance, and reliability. Additionally, since transistors are less reliable with aggressive technology scaling, the combinational logic of ECC hardware is also vulnerable to faults. Thus, we also enhance the router with self-diagnosis function to ensure that the error control circuitries (i.e., ECC encoder and decoder) are fault-secure.

The proposed adaptive error correction hardware is shown in Fig. 7. The proposed error control hardware can be configured as end-to-end CRC, per-hop SECDED, and per-hop DECTED. At runtime, the ECC hardware dynamically deploys the most appropriate error control scheme guided by the DRL-based control policy discussed in Section 4. We reinforce the fault-prone ECC hardware design by proposing a self-diagnosis ECC fault detector to achieve fault-tolerance in both the communication channels and the ECC circuitry. The proposed self-diagnosis detector can verify the correctness of the ECC hardware operations using low-density parity-check (LDPC) codes [22]. As shown in Figs. 7a and 7b, each ECC encoder, decoder, and corrector is assigned to an ECC fault detector. The detector applies LDPC codes to each syndrome vector (the 9-bit Hamming code C_0 to C_8) from the outputs of different ECC hardware. The circuitry details of the proposed ECC fault detector are shown in Fig. 7c. The last OR gate has 12 inputs. The first 9 inputs are the required LDPC input. Because LDPC codes are proven to detect all the error combinations in the 9-bit syndrome vector [23], the proposed detector can detect malfunctions in the ECC hardware due to transient errors. To enhance the tolerance of the ECC hardware to permanent faults, we uniquely add three more inputs to the last OR gate of the error detector. These three bits can indicate the location of the faulty bit in the 9-bit Hamming code. If an error occurs in the same bit repeatedly, the corresponding gates in the ECC hardware will be marked as faulty, and

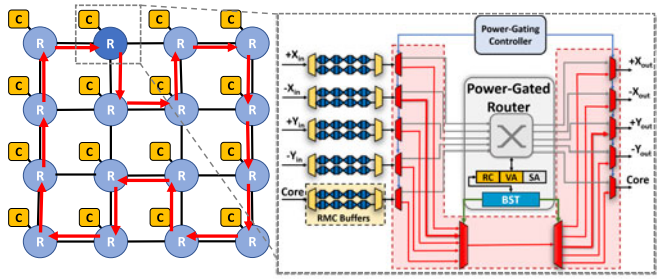

Fig. 8. Microarchitecture of a bypass route in each router. An example of the constructed bypass ring is shown on the left side.

the specific faulty gate will be power-gated, while the bit-level operations will be performed by the other underutilized gates (e.g., the XOR gates for DECTED). Additionally, note that the proposed ECC fault detectors are only activated when the SECDED/DECTED scheme is enabled.

2.4 Router Bypass Route Design

In this paper, we modify the stress-relaxing bypass technique originally proposed in [8], which proactively power-gates and bypasses the NoC router to save power and prevent overheating by adding an extra escape link. All of the inter-router escape links together construct a bypass-ring network to retain connectivity when there are a significant number of failures. The bypass-ring network fully connects each router in the NoC. An example of a chip-level bypass ring in a 4×4 2D mesh is demonstrated with red arrows on the left-hand side in Fig. 8. When all the possible links are faulty in one direction (i.e., all the corresponding RMCs are faulty), the escape link will be activated. Each escape link has one single-flit latch as the link in the stress-relaxing bypass route. The incoming flit will be stored at that single-flit latch and propagated using a round robin scheme. In this way, packets can pass through the faulty router and proceed to the next router. In addition, the proposed design continues to utilize the BST for routing information while the router is bypassed. This retains the connectivity of that direction and eliminates resource starvation to prevent deadlock. In this way, the new bypass route with escape links can further enhance reliability in the presence of faults. Additionally, to reduce the area overhead of the cross-router bypass-ring network, the ring network does not support bypassing in all directions, which means that each network interface can accept packets from only one specific upstream router and forward packets to specific downstream router.

3 Proactive Operation Modes

In this section, we propose ten proactive operation modes (1 power-gating mode and 9 fault-tolerant modes) for CURE routers. Each operation mode has various configurations of RMC, adaptive error control hardware, and power-gating strategies. Each CURE router occasionally and independently selects and deploys an operation mode proactively using a DRL-based control policy (described in Section 4). The operation modes are detailed below.

- *Operation Mode 0 - Power-Gating Mode:* In this mode, the router is power-gated, while the bypass route is enabled. The RMC channel is configured as 2:2 for

both directions, and the RMC buffers are used to store the incoming flits. This operation mode is activated either when the router is underutilized or when a high risk of overheating is predicted. This mode mitigates permanent faults and saves static power.

- *Operation Modes 1, 2, and 3 - CRC-Only Mode:* Operation modes 1, 2 , and 3 have different RMC direction configurations: 2:2 for mode 1, 1:3 for mode 2, and 3:1 for mode 3. However, these operation modes share the same error control configurations. In these modes, the RMCs are configured as storage buffers, and the entire adaptive ECC hardware is power-gated, so that only CRC is enabled. These operation modes are beneficial to save power and eliminate ECC computational overhead when the error level is low.
- *Operation Modes 4, 5, and 6 - Per-hop SECDED Mode:* In these modes, the router's adaptive ECC hardware is partially activated to perform per-hop SECDED. This configuration is beneficial when SECDED can handle most of the faults. Otherwise, it will either lead to unnecessary power and latency penalties (when the error level is low) or excessive re-transmissions (when errors cannot be corrected by SECDED). The RMC buffers are configured as re-transmission buffers. Similar to operation modes 1, 2, and 3, the RMC direction configurations are set to 2:2 for mode 4, 1:3 for mode 5, and 3:1 for mode 6.
- *Operation Modes 7, 8, and 9 - Per-hop DECTED Mode:* In these modes, the router activates the entire adaptive ECC hardware to enable DECTED. This is the situation where the flits are more likely to contain errors of 2 or more bits. The RMC buffers are configured as re-transmission buffers. Similarly, the RMC direction configurations are set to: 2:2 for mode 7, 1:3 for mode 8, and 3:1 for mode 9.

The dynamic selection of operation modes is performed by each router independently in a sequence of discrete time steps using the DRL-based control policy presented in Section 4. At each time step, each router decides which operation mode to apply for the following time step and passes the decision to the downstream router. In this way, the downstream router will be informed to configure the ECC decoder located in the corresponding input port to apply the correct ECC coding, such that it is synchronized with the ECC coding of the upstream router's encoder at output port at the next time step. As demonstrated, the proposed dynamic operation modes allow individual routers to utilize the most suitable technique at runtime, resulting in greater benefits for the entire network.

4 Proposed Deep Reinforcement Learning (DRL)-Based Control Policy

We present a new per-router deep reinforcement learning-based control policy to dynamically select operation modes that can lead to the maximum system-level performance. Reinforcement learning is an online learning algorithm that learns and optimizes the behavior of autonomous RL agents [20](e.g., routers) from the dynamic interactions between the agents and the environment (e.g., NoC system)

Category	Features	Description
Router Input Related Metrics	1. +X link utilization	Input flits/cycle of +X input port
	2. –X link utilization	Input flits/cycle of –X input port
	3. +Y link utilization	Input flits/cycle of +Y input port
	4. –Y link utilization	Input flits/cycle of –Y input port
	5. Local port link utilization	Input flits/cycle of local port
Buffer Related Metrics	6. +X buffer utilization	the buffer utilization of +X input port
	7. –X buffer utilization	the buffer utilization of –X input port
	8. +Y buffer utilization	the buffer utilization of +Y input port
	9. –Y buffer utilization	the buffer utilization of –Y input port
	10. Local port buffer utilization	the buffer utilization of local port
Router Output Related Metrics	11. +X Link utilization	Output flits/cycle of +X input port
	12. –X Link utilization	Output flits/cycle of –X input port
	13. +Y Link utilization	Output flits/cycle of +Y input port
	14. –Y Link utilization	Output flits/cycle of –Y input port
	15. Local port Link utilization	Output flits/cycle of local port
RMC Related Metrics	16. +X RMC Previous Mode	Operation mode at the last timestep
	17. –X RMC Previous Mode	Operation mode at the last timestep
	18. +Y RMC Previous Mode	Operation mode at the last timestep
	19. –Y RMC Previous Mode	Operation mode at the last timestep
Other Metrics	20. Temperature	Local router temperature in ℃

Fig. 9. Router attributes selected in the state vector.

at runtime. Specifically, in CURE, each router (agent), acts as a learner and a decision-maker and interacts with the NoC system (environment), in a sequence of discrete time steps t = 0, 1, 2, 3, and so on. At time step t, the router observes the current *state* by extracting runtime system attributes(e.g., buffer utilization, temperature, etc.), takes an *action* by selecting one of the proposed operation modes and applies it at the next step $t' = t + 1$. Next, at time step t +1, upon taking the action selected in the previous step, the NoC attributes change and result in a new state s'. The new state is fed back to the agent, and the time step is incremented. In addition to observing the new state, the agent also receives a *reward* r. A *policy* π maps states to actions, specifying how to choose actions given the state of the environment to maximize the cumulative reward. For the router, the cumulative reward will be a function of energy, performance, and reliability over the entire sequence of actions. An RL algorithm continually evolves the policy based on the router's past interactions with the NoC system.

Design Space and Action-Value Function. In CURE, the state space S is defined as a vector of several NoC system metrics, or features, listed in Fig. 9. These features contain input-related metrics (attributes 1 to 10), output-related metrics (attributes 11 to 15), RMC related metrics (attributes 16 to 19), and local operation temperature (attribute 20). At each time step, the agent takes an action according to the monitored state. The action space A = $\{a_0, a_1, a_2, \ldots, a_9\}$ contains the ten operation modes from which the routers can select.

The goal of the agent is to optimize its long-term *return*, which is represented by the discounted sum of future rewards. The return at time step *t* is defined as

$$\mathbb{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots. \quad (1)$$

The variable γ (where $0 \leq \gamma \leq 1$) is a *discount rate*, which determines the impact of future rewards on the total return: as γ approaches 1, the agent becomes less near-sighted by giving more weight to future rewards.

In this paper, with the goal of simultaneously improve performance, energy-efficiency, and reliability, we design the reward function for router i at time step t as

$$r_{i,t} = -\log_{\alpha}(Latency_{i,t}) - \log_{\beta}(Power_{i,t}) - \log_{\lambda}(Aging_{i,t}). \quad (2)$$

The *Latency* refers to the average end-to-end latency of the specific router i, *Power* contains both static and dynamic power consumption. Additionally, the aging factor is calculated using the aging model, which is described in detail in Section 5.1. The α, β, and λ is used to emphasize the importance of each individual desired goal. In this paper, we set all three parameters, α, β, and λ, to 1.

Deep Q-Learning Approach. In DRL, a *model* of the environment, specified through a probability distribution $p(s_{t+1}, r_{t+1}|s_t, a_t)$, characterizes how the state of the environment changes as a result of an agent action, and the reward that the agent receives after each action. Correspondingly, DRL agents compute an action-value function $Q^{\pi}(s, a)$ that estimates the return that they are expected to receive in this model of the environment if they start in state s, take action a, and follow the policy π for the remaining actions.

To find the optimal Q-value function $Q^*(s, a)$ that maximizes the expected return, we use the tabular Q-learning algorithm [18]. Q-values are initialized with zeros for all possible (s, a) pairs at the beginning. At each time step, the Q-learning algorithm chooses actions based on the current Q such that, over many time steps, all actions are taken in all states. After taking an action a and observing the reward r and new state s', the action-value entry $Q(s, a)$ is changed using the following temporal difference rule:

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]. \quad (3)$$

The learning rate α can be reduced over time and determines how well Q-learning will converge. It can be shown that for appropriate values of α, Q-learning converges to the optimal Q-value function Q* and its corresponding optimal policy π^* [18]. To explore unvisited regions of the state-action space, an ϵ-greedy policy is also applied to π^*, where agents also have a probability of ϵ to select a random action rather than always taking the action with the maximum Q-value [24].

Note that we intentionally design a reward function with a negative value to achieve a better high-level performance of RL. Because the Q-values of all the visited state-action pairs are negative and all the unvisited pairs have Q-values equal to zero, the RL agent will always select unvisited state-action pairs (zero is greater than any negative number). It allows the RL agents to explore the state-action pairs as much and as fast as possible, which leads to shorter convergence time for optimal decision making.

In conventional RL, the optimal policy π^* is recorded in a state-action mapping table called Q-table, where all Q-values are stored. As mentioned, each state vector consists of a number of features. When the RL agent observes any new state-action pair, it creates a new entry in the Q-table to record its actions and associated Q-values. Although the feature values are discretized into limited bins, the area consumption for the Q-table is excessive. To address this problem, we implement deep Q-learning by replacing the state-action table with an offline-trained artificial neural network (ANN) to reduce the hardware costs. The ANN calculates the state-action value rather than storing the entire state-action table in the router, thus eliminating the storage space for state-action pairs.

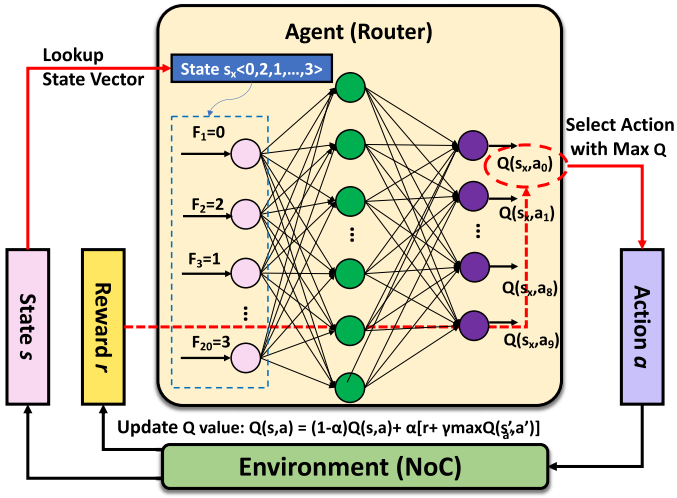

Fig. 10. Q-Learning process. For time step t, the action a_0 with the maximum Q-value of current state s_x is selected. The reward for action a_0 will be calculated after a_0 impacts the NoC environment. The Q-value will be updated following the Q-value updating rule. α is a learning rate , and γ is the discount rate.

In CURE, each ANN consists of 20 neurons at the input layer, 30 neurons at the hidden layer, and 10 neurons at the output layer. For training, we use a cycle-accurate network simulator to simulate the dynamic interactions and the online process of reinforcement learning. At this stage, Q-values of different state-action pairs are explored using real-world applications. All the visited state vectors and their corresponding Q-values (calculated with the reward function) are recorded as training sets for the ANN. Specifically, the attribute values of the state vector are ANN inputs, and the Q-values are the desired outputs. During each time step, similarly to updating the Q-value using reward values in conventional RL, the ANN follows the Q-value update rule and record ΔQ as the output error. Then, it uses mini-batch gradient descent to back propagate this error to the hidden layer to tune the weights. The training time is not counted to the timing overhead of the proposed NoC system. During the testing (or inference) phase, the ANN monitors the attribute values and uses the input values and weights to calculate 10 Q-values. The action with the highest Q-value will be selected.

The timing and area overheads of the proposed deep Q-learning are discussed in Section 6.4.

The Working of the Proposed DRL-based Controller. Fig. 10 demonstrates the working of the DRL-based control logic when running a benchmark. At each time step, the process goes through several stages. In the first stage, the router uses the feature values $F_1, F_2, \ldots, F_{20}$ in the state vector s as inputs of the ANN. In Fig. 10, we assume current state s matches state s_x in the mapping table. In the second stage, the ANN calculates the Q-values of all possible state-action pairs in the current state entry, and the router selects an action a, which has the maximum $Q(s, a)$-value for the next time step (we assume that a_0 in Fig. 10 has the maximum Q-value). Upon taking the action a, the NoC system transits to a new state s'. In the last stage of the current time step, the

TABLE 1
Simulation Environment Setup

# of cores	64 out-of-order CPUs @ 32 nm
Voltage and Frequency	1.0 Volt, 2.0 GHz
NoC Parameters	8 × 8 2D Mesh, 4-stage routers
Packet Size	4 flits
Flit Size	128 bits
Cycle Delay	4 cycle to L1 cache 8 cycle to L2 cache 160 cycle to main memory
Buffer Numbers* of Different Technologies	4RB-4VC-0CB (SECDED, SHIELD, Vicis) Avg. 8CB, 2 Directions (QORE) 2RB-4VC-8CB (IntelliNoC) Avg. 2RB-4VC-8CB, 2 Directions (CURE)

* *RB: router buffer, VC: virtual channel, and CB: channel buffer.*

NoC system provides a reward r (defined in 2) to the router. The reward will be used to update $Q(s, a)$. Each router will repeat all of these stages at each time step.

The initialization of the per-router controller and the state-action mapping table will be discussed in Section 5.2.

5 Evaluation Methodology

In this section, we present the fault injection model used in our experiments to inject timing errors into NoC, the simulation setup, and the benchmarks that we use. Faults that occur in the routing table and machine learning module are beyond the scope of this paper.

5.1 Fault Injection Models

To quantify the reliability improvement of the proposed designs for timing errors, we first create a transient error injection model to realistically produce a probability of timing errors for each link. The proposed transient fault injection model is a combination of the existing VARIUS [25] fault model and HotSpot [26] thermal model. At runtime, first, HotSpot uses the values of router supply voltage, operation frequency, and link utilization to obtain router operating temperature at runtime. The temperature values are fed into the VARIUS transient error model to generate the probability of timing errors (R_e) for each transmitted bit. Using R_e, the probability of which n-bit flit is faulty can be calculated as follows:

$$P_{fault} = 1 - (1 - R_e)^n. \tag{4}$$

To assess the reliability improvement for permanent faults, we utilize the permanent fault model proposed in [8]. Specifically, we model and calculate the *aging* factor in (2) by correlating the shift in the threshold voltage of the transistor (ΔV_{th}). The ΔV_{th} is calculated using both ΔV_{th_NBTI} given by [27], [28] and ΔV_{th_HCI} given by [29], [30].

We model the *aging* factor given in (2) as follows:

$$\begin{cases} \Delta V_{th} = \Delta V_{th\_NBTI} + \Delta V_{th\_HCI} \\ Aging = 1 + \frac{\Delta V_{th}}{V_{th0}} \times 100\% \end{cases}. \tag{5}$$

Note that the *aging* factor is designed to have a value greater than 1 such that it can be used in the reward function.

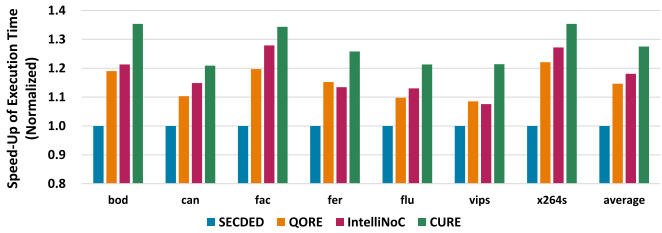

Fig. 11. Speed-up of full application execution time comparison, normalized to the SECDED baseline (higher is better).

5.2 Simulation Setup

We evaluate our proposed architecture using a modified version of the cycle-accurate network simulator *Booksim2* [21]. We also use Netrace [31] to capture cycle-accurate benchmark traces for the network simulator. Table 1 shows the simulation setup.

As mentioned, DRL parameters (α, γ, and ϵ) and ANN setups (e.g., neuron numbers) can impact the performance of DRL [32], [33]. In this paper, we set α, γ and ϵ to 0.1, 0.9 and 0.05, respectively. The operation modes of all routers are initialized to mode 1.

Workloads from PARSEC benchmarks [34] are tested. The benchmarks from PARSEC are transformed into trace files that contain trace format packet injection/ejection events and offer runtime information (such as time, packet size, transmission source, destination, and event type). Because CURE is the first NoC design framework that is fault-secure to both link failure and router failure, it is difficult to find a single state-of-the-art technology to compare with. Therefore, we conduct two sets of experiments to evaluate the performance of CURE in Section 6. In Section 6.1, we compare the performance of the proposed RL framework to the performances of the following state-of-the-art techniques: a static baseline using SECDED, QORE [7], and IntelliNoC [8] while link failures are injected. In Section 6.2, we compare CURE with SHIELD [3] and Vicis [4] while intra-router permanent faults are injected. For the RL-based IntelliNoC and DRL-based CURE, we train the per-router policy using a subset of PARSEC (blk, dedup, fre, and swa). Then, we use the remaining applications in PARSEC to test performance. The testing phase for each benchmark lasts a full application execution time. The control policy is dynamically updated by applying the temporal difference rule (3) every 1,000 cycles.

6 Evaluation Results and Analysis

6.1 Performance Analysis With Faulty Links

In this subsection, system-level performance metrics are evaluated when link failures are injected. Before runtime, permanent faults are randomly inserted into a percentage of links with a probability of 5 percent. At runtime, transient errors are injected into the links using the error injection model presented in Section 5. The evaluation results are presented below.

Execution Speed-up. The speed-up is obtained by calculating the ratio of the full application execution time of various techniques (baseline, QORE, and IntelliNoC) to the execution time using the proposed CURE technique running different benchmarks. The speed-up comparison is shown in Fig. 11. As shown in Fig. 11, CURE achieves the largest

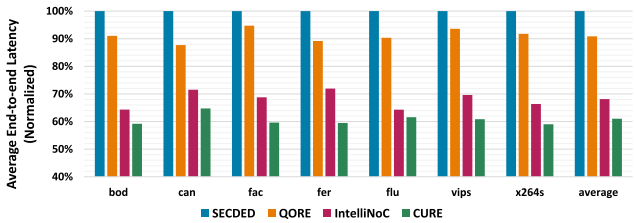

Fig. 12. Network latency comparison using average end-to-end latency, normalized to the SECDED baseline (lower is better).

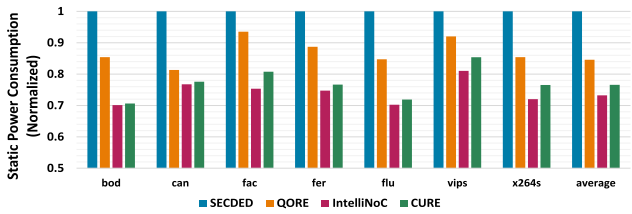

Fig. 14. Overall static power consumption comparison, normalized to the SECDED baseline (lower is better).

speed-up of all evaluated techniques. QORE achieves a 14 percent speed-up over the SECDED baseline since the dual-direction links can improve the NoC throughput by dynamically allocating the limited link bandwidth for unbalanced traffic. IntelliNoC results in an average speed-up of 18 percent due to the ability to reduce ECC overhead and retransmission traffic. However, in some benchmarks (i.e., fer and vips), IntelliNoC achieves worse performance than QORE. This is because IntelliNoC's fixed-direction MFAC can be the bottleneck when multiple link failures occur on the same channel. However, because CURE does not have such limitations, it successfully accelerates benchmark execution by 27 percent.

Network Latency. We define network latency as the average end-to-end latency of all transmitted packets of the full execution of each benchmark application. To measure the end-to-end latency, first, when a packet is injected from the source node, the injection time is recorded. Second, when the packet reaches the destination node and accepted by the destination node (after passing the error checking), the packet acceptance time is also recorded. The end-to-end latency for that packet is the time difference between injection time and acceptance time, which is recorded using the number of clock cycles. Fig. 12 shows the normalized network latency for different techniques. CURE achieves an average of end-to-end latency reduction of 39 percent. Note that QORE only achieves a 9 percent latency reduction over the baseline due to the timing overhead of channel buffer allocation and static error control scheme. Techniques with RL-based policies (i.e., IntelliNoC and CURE) both achieve latency reductions of over 30 percent, which implies that dynamic proactive policy such as the RL-based policy can minimize re-transmission and thus improve overall latency.

Energy-Efficiency. We define energy-efficiency as

$$Energy - E\ f\ ficiency = \left[\left(P_{static} + P_{dynamic}\right) \times T_{exec}\right]^{-1}. \tag{6}$$

P_{static} and $P_{dynamic}$ are static and dynamic power consumption, respectively. We first model the static power with Synopsys Library Compiler for the designed NoC. Since Synopsys cannot evaluate the dynamic power accurately for different benchmark applications, we fed the static power parameters captured by Synopsys to DSENT [35] power model. During application execution, DSENT calculates the average dynamic power by the number of buffer writes, crossbar, and VA/SA activities within full application execution time. T_{exec} is the benchmark execution time. Fig. 13 shows the energy-efficiency measurements for all techniques studied and normalized to the SECDED baseline. CURE improves energy-efficiency by 92 percent compared to the baseline, while the energy-efficiency improvements using QORE and IntelliNoC are 36 and 77 percent, respectively.

Overall Static Power Consumption. Fig. 14 shows the overall static power consumption for the various techniques. QORE reduces static power consumption by 11 percent due to the elimination of router buffers. IntelliNoC achieves a static power reduction of 27 percent thanks to its power-gating and bypass scheme. Similar to IntelliNoC, the use of power-gating and bypass is beneficial in CURE. However, due to the extra logic and circuitry in the modified ECC and pipeline stages, the proposed CURE only achieves a static power reduction of 24 percent.

Overall Dynamic Power Consumption. As discussed, dynamic power consumption can be lowered by reducing the number of router buffers and/or by mitigating faults and reducing the number of re-transmissions. CURE, using RMC and dynamic error control, is able to significantly reduce re-transmission traffic. Consequently, CURE outperforms all other techniques in reducing dynamic power consumption as shown in Fig. 15.

6.2 Performance Analysis With Intra-Router Failures

In this subsection, permanent faults are injected into the router circuit (including VA, SA, crossbar, and ECC hardware). Before runtime, faults are randomly inserted into a

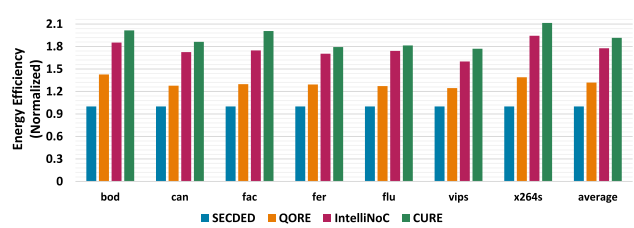

Fig. 13. Energy-efficiency comparison, normalized to the SECDED baseline (higher is better).

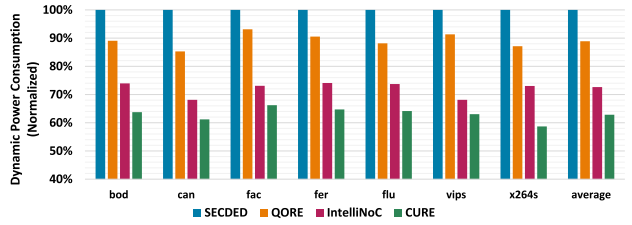

Fig. 15. Overall dynamic power consumption comparison, normalized to the SECDED baseline (lower is better).

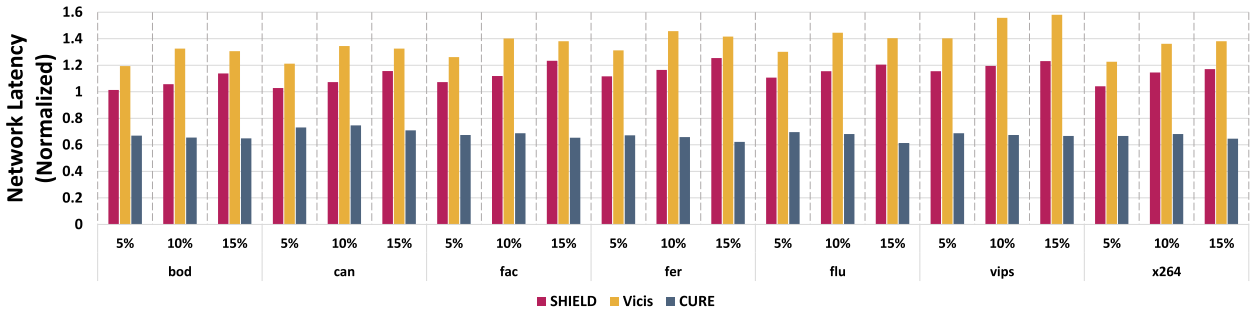

Fig. 16. Network latency comparison using average end-to-end latency, normalized to the SECDED baseline (lower is better). Each benchmark is tested under 5, 10, and 15 percent permanent fault rates in router.

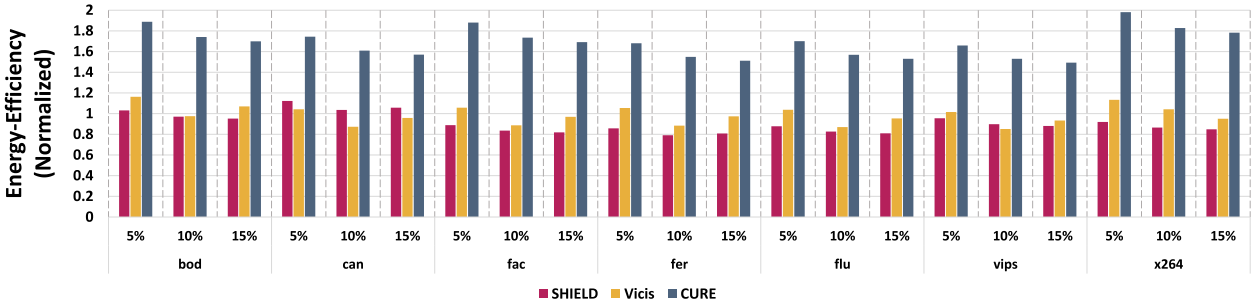

Fig. 17. Energy-efficiency comparison, normalized to the SECDED baseline (higher is better). Each benchmark is tested under 5, 10, and 15 percent permanent fault rates in router.

percentage of the router components ranging from 5 to 15 percent. At runtime, transient faults are injected only to the links using the proposed error injection model. The transient fault-handling technique used by SHIELD and Vicis is static SECDED, while CURE utilizes the proposed adaptive error control hardware. The network latency and energy efficiency of different techniques are evaluated, and the results are normalized to the static SECDED baseline. Because the SECDED baseline cannot tolerate router failures, the intra-router error rate is set to 0 percent for the baseline.

Network Latency. Fig. 16 shows the normalized network latency for different techniques at various error rates. The proposed CURE framework achieves at least 30 percent network latency reduction under 5, 10, and 15 percent router failure rates. However, both of the other fault-tolerant techniques incur excessive network latency.

Energy-Efficiency. Fig. 17 shows the normalized energy-efficiency for different techniques. The energy-efficiency of all the techniques tends to decrease as the error rate increases. However, there are some exceptions to Vicis. When the error rate exceeds 10 percent, Vicis re-configures itself less often, which can significantly reduce execution time. As shown in the figure, the proposed CURE framework achieves the highest energy-efficiency among all the other techniques. This result implies that the use of channel buffers and adaptive error control hardware can substantially reduce power and execution time.

6.3 Reliability Improvement Analysis

We use mean-time-to-failure (MTTF) to evaluate the reliability of the proposed design for permanent faults [36], [37]. Fig. 18 shows the normalized MTTF comparison. As shown in Fig. 18, the proposed CURE framework is 7.7× more reliable than the baseline router which is unprotected from permanent faults, while the highest normalized MTTF value of other fault-tolerant designs is 4.6× (SHIELD). This improvement is achieved thanks to the reduced operation temperature by the RL-based control policy and the modified pipeline and ECC hardware in each router.

6.4 Overhead Analysis

We evaluated the area overhead of each technique with Synopsys and 32 nm technology library with the supply voltage set to 1.0 Volt, and clock frequency set to 2.0 GHz. The area overhead is shown in Table 2.

As shown in this table, IntelliNoC, QORE, and CURE consume less area than the baseline due to the use of link storage. Due to the additional circuitry design, Vicis has the largest area overhead. CURE and IntelliNoC incur additional area overheads thanks to the RL-based controlling modules. The area overhead of the DRL-based controller

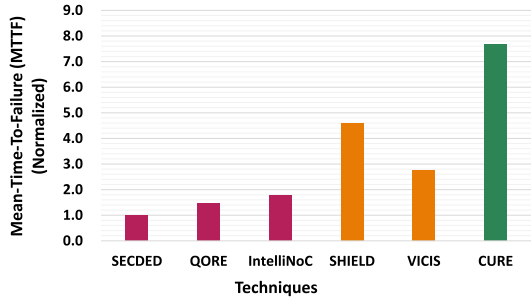

Fig. 18. Mean-time-to-failure (MTTF) comparison, normalized to the SECDED baseline (higher is better).

TABLE 2
Area Overhead Comparison (μm^2)*

	Baseline	QORE	SHIELD	Vicis	IntelliNoC	CURE
Router Buffer	1248.3 ×16 /port	-	1248.3 ×16 /port	1248.3 ×16 /port	1248.3 ×8 /port	1248.3 ×8 /port
Control logic & Crossbar	9004.7	9662.4	12066.3	41712.0	9004.7	11613.6
Channel	136.7	5948.4	136.7	136.7	2869.6	2974.2
ECC	3325.4	3325.4	3325.4	3325.4	3940.3	4058.1
Total	119807.0	79764.4	83953.1	152514.3	89313.7	89256.2
%Change	-	-33.4%	+2.6%	+27.3%	-25.4%	-25.5%

includes the area consumption of both ALUs (adder, multiplier, and Sigmoid function) and SRAM for calculating, updating, and storing the ANN. The simulation shows that the area overhead for ALUs and SRAM are 992.2μm^2 and 838.6μm^2, respectively. This implies 0.8 and 0.7 percent of the area consumption of the baseline router. Therefore, the area cost is reduced as compared to the 4 percent area overhead of table-based RL in previous designs. Moreover, CURE requires additional timing overhead for calculating the Q-values using ANN. Using [38], it shows that at each time step, the timing overhead of CURE is estimated to be 160 ns. Using the similar method as [39], this latency can be overlapped by a large time step. Specifically, we use two sets of different intervals for monitoring the attributes and the controlling to minimize the negative effect of this latency. The two sets of intervals are offset by the ANN computation time, which can pipeline the overhead effectively. By doing so, the ANN control computing does not block either the monitoring process or the controlling. Therefore, the use of ANN will not negatively impact the overall performance metrics. Additionally, the timing overheads for the MFAC (IntelliNoC) and RMC (CURE) configurations are both estimated around 45 to 50 cycles. Furthermore, The control logic of the proposed DRL consumes an additional 0.26 mw power, which implies 4 percent of the static power consumption of the baseline router.

6.5 Sensitivity Study

Impact of Input Data Size. In this test, we explore the impact of different input data sizes on network latency and energy-delay product (EDP), and the blackscholes application in the PARSEC benchmark is used. Lower network latency and EDP indicate better system performance. The evaluation result illustrated in Fig. 19 shows that as input data size increases, the performance metrics remain the same. This is because the traffic patterns, network intensity, and workload allocations for the same benchmark processing phase are the same, for different data input sizes. However, by reducing the NoC size to a 4×4 mesh, which incurs a higher network intensity, the proposed adaptive ECC hardware can achieve better network performance, as compared to the static SECDED baseline.

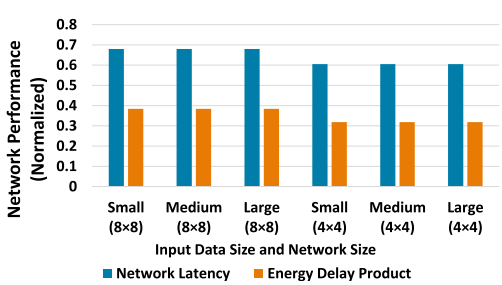

Fig. 19. Impact of different input data sizes and different NoC sizes.

Impact of Time Step Length. In this test, we varied the time step t starting from 200 to 10,000 clock cycles. The evaluation results using the blackscholes application are illustrated in Fig. 20a. As shown in Fig. 20a, a longer cycle time (10K cycles) has a negative impact on performance due to coarse-grain control. Meanwhile, aggressively reducing the length of time steps (200 clock cycles) also leads to a degradation in performance because the computational overhead of DRL-based control policy will dominate performance.

Impact of Discount Rate γ. Fig. 20b indicates the impact of the discount rate γ on network latency and EDP. The blackscholes application is used in this test. As shown in Fig. 20b, the network latency and EDP initially improve with larger γ, yet aggressively increasing γ can also lead to Q-learning failing to converge, which negatively affects the system performance. The best performance is achieved when γ equals 0.9.

Impact of Exploration Probability ϵ. Fig. 20c shows the impact of ϵ values on network latency and EDP, using the blackscholes application. When ϵ is 0, the router always selects the initial mode most of the time. As ϵ increases from 0 to 1, the router tends to explore new state-action pairs more frequently. Further, when ϵ equals 1, the router will take actions entirely at random. As shown in Fig. 20c, the best system performance is achieved when ϵ equals 0.05.

Impact of the Hidden Layer Size of the ANN. Because the number of neurons used in the hidden layer of the ANN can affect the accuracy of calculating $Q(s, a)$, thereby impacting the decision making of the DRL controller, we vary the size of

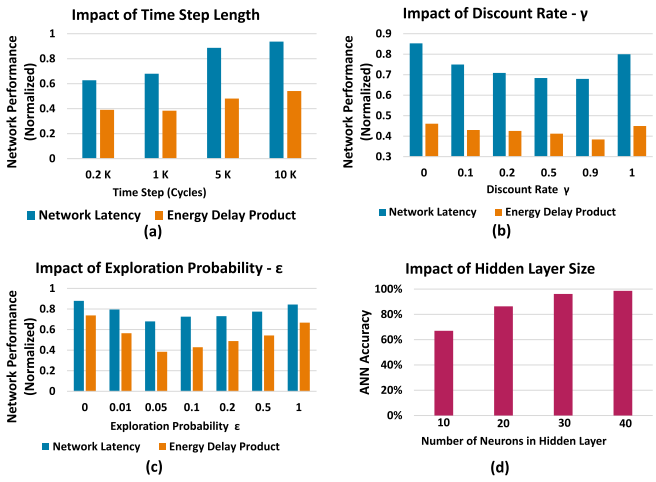

Fig. 20. Impact of (a) time step length, (b)discount rate γ, (c) exploration probability ϵ, and (d) hidden layer size of of DQL on network performance metrics. Results are normalized to the baseline.

the hidden layer to study its impact on calculation accuracy, as shown in Fig. 20d. Fig. 20d shows that the calculation accuracy improves as the hidden layer size increases. To save area consumption and reduce the timing overhead of the ANN, we select the hidden layer of 30 neurons.

7 Related Works

There has been considerable work in improving the energy-efficiency and reliability in NoCs. In the following, we briefly highlight some of the directly relevant works.

Power-Saving Designs for NoCs. Because static power consumption has become a substantial portion of overall network power, power-gating (PG) techniques that power off underutilized network components have been shown to be effective for static power savings [14], [15], [40]. However, conventional power-gating schemes for routers tend to substantially increase network latency due to a reduced number of active routers in the network and extra control overhead in managing power-gating. Another approach proposed for reducing network power is reducing router buffers. Michelogiannakis and Dally [11] and Kodi *et al.* [7], [12]have shown that eliminating router buffers is beneficial for both static and dynamic power reduction. However, simply replacing router buffers with channel buffers leads to penalties in network congestion and latency [7], [11], [12].

Fault-Tolerant Designs for NoCs. In NoC, both transient and permanent faults can manifest during transmission. CRC [6] is a basic transient fault detection technique that is often used for NoCs. Flits are encoded by a local CRC encoder in the router before transmission, and are decoded by the destination CRC decoder to perform error detection. If the destination router detects errors, a re-transmission request is sent to the source router to re-transmit the flit. To mitigate transient faults, per-hop error correction codes (ECCs) are generally deployed. SECDED is one of the most commonly used ECC techniques in NoCs [4]. To handle permanent faults caused by transistor aging [29], a number of techniques have been proposed using load-balancing [7], circuitry redundancy [37], and adaptive routing techniques [4] among others. Note that most of the techniques are static in nature, with CRCs or SECDEDs being deployed all the time, regardless of whether faults are present. Reliability-enhancement mechanisms based on static techniques have been shown to require excessive power consumption, and longer delays, thereby significantly degrading NoC performance [17], [37], [41].

Learning-Enabled NoC Designs. Multiple machine learning techniques have been introduced to balance design trade-off or predict traffic in NoCs. These works target achieving high power efficiency [8], [16], [24], [39], [42], [43], [44], [45], and enable fault-tolerant design [5], [8], [45]. For example, [44] discovers that the wake-up latency of PG and performance degradation of using DVFS are the bottlenecks of implementing PG and DVFS simultaneously. Hao *et al.* [44] have shown that applying machine learning to handle the dynamic trade-offs of DVFS and PG can achieve optimal power savings. [45] introduces a proactive fault-tolerant mechanism to optimize energy efficiency and performance with reinforcement learning (RL). Further, [8] proposes to use channel buffers to achieve higher power savings in addition to [45]. We extend these existing works by proposing link reversibility, adding hard error tolerance, and using DRL to reduce control overhead.

8 Conclusion

In this paper, we propose *CURE*, a learning-based NoC design that can simultaneously improve performance, energy-efficiency, and reliability. CURE consists of reversible multi-function adaptive channel, enhanced fault-tolerant router circuitry, ten unique operation modes, and a DRL-based dynamic control policy. With DRL, each router learns from the NoC behavior and updates a control policy to select an optimal operating mode at any given time. The experimental results illustrate that CURE decreases network latency by 39 percent, improves energy efficiency by 92 percent over the static SECDED baseline. Using the mean-time-to-failure metric, we show that the proposed framework is 7.7× more reliable than the baseline NoC architecture.

Acknowledgments

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

References

[1] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.
[2] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Annu. Des. Autom. Conf.*, 2001, pp. 684–689.
[3] P. Poluri and A. Louri, "Shield: A reliable network-on-chip router architecture for chip multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 3058–3070, Oct. 2016.
[4] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A reliable network for unreliable silicon," in *Proc. 46th ACM/EDAC/IEEE Annu. Des. Autom. Conf.*, 2009, pp. 812–817.
[5] D. DiTomaso, T. Boraten, A. Kodi, and A. Louri, "Dynamic error mitigation in NoCs using intelligent prediction techniques," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2016, pp. 1–12.
[6] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. NJ, USA: Prentice Hall, 2004.
[7] 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. 20th Int. Symp. High Perform. Comput. Archit.*, 2014, pp. 320–331.
[8] 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. 46th Int. Symp. Comput. Archit.*, 2019, pp. 589–600.
[9] K. Aisopos, A. DeOrio, L.-S. Peh, and V. Bertacco, "ARIADNE: Agnostic reconfiguration in a disconnected network environment," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, 2011, pp. 298–309.
[10] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 4–15, 2006.
[11] G. Michelogiannakis and W. J. Dally, "Elastic buffer flow control for on-chip networks," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 295–309, Feb. 2013.
[12] A. K. Kodi, A. Sarathy, and A. Louri, "iDEAL: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures," *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 241–250, 2008.
[13] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Ultra fine-grained run-time power gating of on-chip routers for CMPs," in *Proc. 4th ACM/IEEE Int. Symp. Netw.-on-Chip*, 2010, pp. 61–68.

[14] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 320–331, 2013.
[15] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit.*, 2013, pp. 508–519. [Online]. Available: http://dx.doi.org/10.1109/HPCA.2013.6522345
[16] M. Clark, A. Kodi, R. Bunescu, and A. Louri, "LEAD: Learning-enabled energy-aware dynamic voltage/frequency scaling in NoCs," in *Proc. 55th ACM/ESDA/IEEE Des. Autom. Conf.*, 2018, pp. 82:1–82:6. [Online]. Available: http://doi.acm.org/10.1145/3195970.3196068
[17] 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.
[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[19] L. Busoniu, R. Babuska, and B. De Schutter, "Multi-agent reinforcement learning: A survey," in *Proc. 9th Int. Conf. Control Autom. Robot. Vis.*, 2006, pp. 1–6.
[20] V. François-Lavet *et al.*, "An introduction to deep reinforcement learning," *Found. Trends® Mach. Learn.*, vol. 11, no. 3/4, pp. 219–354, 2018.
[21] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2013, pp. 86–96.
[22] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for nanomemory applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 4, pp. 473–486, Apr. 2009.
[23] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.
[24] Q. Fettes, M. Clark, R. Bunescu, A. Karanth, and A. Louri, "Dynamic voltage and frequency scaling in NoCs with supervised and reinforcement learning techniques," *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 375–389, Mar. 2019.
[25] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, Feb. 2008.
[26] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.
[27] M. A. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Rel.*, vol. 45, no. 1, pp. 71–81, 2005.
[28] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," in *Proc. Custom Integr. Circuits Conf.*, 2006, pp. 189–192.
[29] D. Lorenz, G. Georgakos, and U. Schlichtmann, "Aging analysis of circuit timing considering NBTI and HCI," in *Proc. 15th IEEE Int. On-Line Testing Symp.*, 2009, pp. 3–8.
[30] H. Kim, A. Vitkovskiy, P. V. Gratz, and V. Soteriou, "Use it or lose it: Wear-out and lifetime in future chip multiprocessors," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2013, pp. 136–147.
[31] J. Hestness, B. Grot, and S. W. Keckler, "Netrace: Dependency-driven trace-based network-on-chip simulation," in *Proc. 3rd Int. Workshop Netw. Chip Archit.*, 2010, pp. 31–36.
[32] E. Ipek, O. Mutlu, J. F. Martínez, and R. Caruana, "Self-optimizing memory controllers: A reinforcement learning approach," *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 39–50, 2008.
[33] Y. Bai, V. W. Lee, and E. Ipek, "Voltage regulator efficiency aware power management," in *Proc. 22nd Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2017, pp. 825–838.
[34] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *Proc. 5th Annu. Workshop Model. Benchmarking Simul.*, 2009, vol. 2011, p. 37.
[35] C. Sun *et al.*, "DSENT-A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Proc. IEEE/ACM 6th Int. Symp. Netw.-on-Chip*, 2012, pp. 201–210.
[36] J. Shin, V. Zyuban, Z. Hu, J. A. Rivers, and P. Bose, "A framework for architecture-level lifetime reliability modeling," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2007, pp. 534–543.
[37] K. Constantinides *et al.*, "BulletProof: A defect-tolerant CMP switch architecture," in *Proc. 12th Int. Symp. High-Perform. Comput. Archit.*, 2006, pp. 5–16.
[38] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2014, pp. 10–14.
[39] J.-Y. Won, X. Chen, P. Gratz, J. Hu, and V. Soteriou, "Up by their bootstraps: Online learning in artificial neural networks for CMP uncore power management," in *Proc. 20th Int. Symp. High Perform. Comput. Archit.*, 2014, pp. 308–319.
[40] L. Chen and T. M. Pinkston, "NoRD: Node-router decoupling for effective power-gating of on-chip routers," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2012, pp. 270–281.
[41] M. Koibuchi, H. Matsutani, H. Amano, and T. M. Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip," in *Proc. 2nd ACM/IEEE Int. Symp. Netw.-on-Chip*, 2008, pp. 13–22.
[42] S. Van Winkle, A. K. Kodi, R. Bunescu, and A. Louri, "Extending the power-efficiency and performance of photonic interconnects for heterogeneous multicores with machine learning," in *Proc. 24th IEEE Int. Symp. High-Perform. Comput. Archit.*, 2018, pp. 480–491.
[43] D. DiTomaso, A. Sikder, A. Kodi, and A. Louri, "Machine learning enabled power-aware network-on-chip design," in *Proc. Des. Autom. Test Eur. Conf.*, 2017, pp. 1354–1359.
[44] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *Proc. 56th Annu. Des. Autom. Conf.*, 2019, pp. 47:1–47:6. [Online]. Available: http://doi.acm.org/10.1145/3316781.3317768
[45] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "High-performance, energy-efficient, and fault-tolerant network-on-chip design using reinforcement learning," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2019, pp. 1166–1171.

Ke Wang (Student Member, IEEE) received the BS degree in electrical engineering from Peking University, Beijing, China, in 2013, and the MS degree in electrical engineering from Worcester Polytechnic Institute, Worcester, Massachusetts, in 2015. He is currently working toward the PhD degree in computer engineering in the School of Engineering and Applied Science, George Washington University, Washington, DC. His research work focuses on optimized NoC design of high performance, power efficiency, and reliability using machine learning.

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, Washington, DC, 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 in 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*.