IntelliNoC: A Holistic Design Framework for Energy-Efficient and Reliable On-Chip Communication for Manycores

Ke Wang* Ahmed Louri* Avinash Karanth§ Razvan Bunescu§
*Department of Electrical and Computer Engineering, George Washington University, Washington, D.C.
§School of Electrical Engineering and Computer Science, Ohio University, Athens, Ohio
{cory,louri}@gwu.edu,{karanth,bunescu}@ohio.edu

ABSTRACT

As technology scales, Network-on-Chips (NoCs), currently being used for on-chip communication in manycore architectures, face several problems including high network latency, excessive power consumption, and low reliability. Simultaneously addressing these problems is proving to be difficult due to the explosion of the design space and the complexity of handling many trade-offs. In this paper, we propose **IntelliNoC**, an intelligent NoC design framework which introduces architectural innovations and uses reinforcement learning to manage the design complexity and simultaneously optimize performance, energy-efficiency, and reliability in a holistic manner. IntelliNoC integrates three NoC architectural techniques: (1) multi-function adaptive channels (MFACs) to improve energy-efficiency; (2) adaptive error detection/correction and re-transmission control to enhance reliability; and (3) a stress-relaxing bypass feature which dynamically powers off NoC components to prevent overheating and fatigue. To handle the complex dynamic interactions induced by these techniques, we train a dynamic control policy using Q-learning, with the goal of providing improved fault-tolerance and performance while reducing power consumption and area overhead. Simulation using PARSEC benchmarks shows that our proposed **IntelliNoC** design improves energy-efficiency by 67% and mean-time-to-failure (MTTF) by 77%, and decreases end-to-end packet latency by 32% and area requirements by 25% over baseline NoC architecture.

CCS CONCEPTS

• **Computer systems organization** → **Interconnection architectures**; **Multicore architectures**; • **Hardware** → **Network on chip**; • **Theory of computation** → Reinforcement learning; • **Networks** → Network performance analysis.

KEYWORDS

Network-on-Chip (NoC), Reinforcement Learning, NoC Performance, Reliability, Energy-Efficiency

ACM Reference Format:
Ke Wang* Ahmed Louri* Avinash Karanth§ Razvan Bunescu§. 2019. IntelliNoC: A Holistic Design Framework for Energy-Efficient and Reliable On-Chip Communication for Manycores. In *The 46th Annual International Symposium on Computer Architecture (ISCA '19), June 22–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3307650.3322274

ISCA '19, June 22–26, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6669-4/19/06…$15.00
https://doi.org/10.1145/3307650.3322274

1 INTRODUCTION

Network-on-Chips (NoCs) [1, 2] have emerged as the standard interconnect fabric for connecting cores, caches, and memory controllers on the chip. With aggressive scaling of transistor technology to lower dimensions (< 10nm), NoC architectures are facing several urgent challenges including increased power consumption (both static and dynamic power), waning reliability (transient and permanent faults), and increased latency.

There has been a significant amount of work recently devoted to improving energy-efficiency, reliability and performance for NoCs [3–11]. Dynamic energy savings have been obtained using voltage and frequency scaling [7, 8], elastic/channel buffers [9, 10], and dynamic buffer allocation [11] among many other techniques. With technology continuing to scale, leakage power has become dominant, and therefore, power-gating techniques with bypass or multiple sub-networks are proposed to take advantage of idle router periods [7, 12, 13] to reduce power consumption. However, aggressive energy saving techniques have negative effects on performance, notably increased latency or execution time since packets are delayed or routes are disconnected [14]. Moreover, with aggressive transistor scaling comes reliability issues in both transient and permanent faults. These faults are handled using many techniques, including adaptive routing, relaxed transmission, forward error detection/correction and packet re-transmission schemes, all of which result in substantial power consumption and longer latency [3, 4, 6]. There is a strong need for a holistic NoC design methodology that simultaneously tackles the challenges of lowering power, increasing performance, and improving reliability, but, as demonstrated, this presents many trade-offs and difficulties.

Designing such an integrated framework is challenging, since it requires the prediction of future NoC behavior and deploying a large number of optimization techniques, which can conflict and offset each other's desired goals. The problem is compounded by the fact that most on-chip applications' traffic patterns, application workloads, component stress, and failures are sometimes unpredictable. Such uncertainty of prediction can result in inadequate decision making, which negatively impacts performance. Moreover, manually designing the rules and strategies for making proactive decisions in NoCs requires substantial engineering efforts and resources, which often result in sub-optimal solutions. This motivates us to explore machine learning techniques to manage the complexity

and automatically learn an optimal control policy to achieve our goal of balancing power, reliability and performance of NoCs. To this end, we propose **IntelliNoC**, a reinforcement learning (RL) [15, 16] design framework for NoCs, which explores the dynamic interactions among NoC components and system-level metrics, and evolves optimal per-router control policies. The major contributions of the paper are as follows:

- **Inter-Router Link Design:** We propose multi-function adaptive channel (MFAC) buffers to be used as inter-router link buffers in IntelliNoC architecture. MFAC buffers can assume several functions: (1) regular repeaters for flit transmission, (2) buffers for storage on the link itself, (3) re-transmission buffers for fault-tolerance, and (4) relaxed timing buffers for reliability and congestion requirements. The MFACs are extensions of the dual-function channels originally proposed in iDEAL [10]. With additional storage available on the inter-router channel, dynamic energy for on-chip storage is reduced at high network loads without any performance degradation. MFACs also provide extra flexibility for improving reliability for both transient (via re-transmission buffers) and permanent faults (via relaxed timing buffers).
- **Router Bypass for Power Savings and Robustness:** We propose a simple router bypass switch, named "stress-relaxing router bypass", that routes flits from upstream MFACs to downstream MFACs without accessing the router buffers or crossbar. In doing so, IntelliNoC saves static power and reduces the stress on the router. Power-gating is deployed at low traffic load to save static power, and additional router bypass is introduced to allow sporadic flits to continue being transmitted without waking up the powered-off router. This has the benefit of saving power and avoiding wake-up latency. Further, at medium-to-high traffic load, the bypass route can be proactively enabled to reduce the stress on the router. This lowers the operating temperature and consequently mitigates the aging effects of the router.
- **Improved Router Microarchitecture Design:** NoC robustness is significantly improved by enhancing the router architecture with per-router adaptive error correction/detection hardware. Both transient and permanent faults are mitigated by this additional hardware along with the router bypass mechanism described above. Per-router adaptive 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 overhead.
- **RL-based Control Policy Design:** We propose several operation modes for the router along with an RL based control policy to handle the dynamic interactions and optimize the trade-offs. The operation modes are intended to maximize power savings and enhance reliability without incurring major performance degradation. Per-router RL agents observe and learn from the entire NoC environment at runtime. The RL agents eventually evolve optimal per-router control policies, which automatically select the optimal operation modes at any given time.

We evaluate the performance of the proposed IntelliNoC architecture using a modified Booksim2 [17] simulator with PARSEC benchmarks on an 8×8 2D Mesh architecture. We show that the proposed **IntelliNoC** provides significant power savings, enhanced reliability, higher performance, and lower area overhead, as compared to traditional NoCs with static fault-tolerant mechanisms.

2 RELATED WORK

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

Improving NoC Energy-Efficiency: As static power consumption has become a substantial portion of overall network power, power-gating (PG) techniques which power off under-utilized network components have been shown effective for static power savings [12, 13, 18]. 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 proposed approach for reducing network power is reducing router buffers. Previous research [9, 10] have shown that eliminating router buffers is beneficial for both static and dynamic power reduction. Elastic Buffers (EB) [9] replace router buffers with flip-flops in inter-router channels, and iDEAL [10] uses three-state repeaters to store data in the inter-router channels. However, simply replacing router buffers with channel buffers leads to penalties in network congestion and latency as show in [9, 10]. Sub-networks (in EB) [9] and dynamic buffer allocation schemes (in iDEAL) [10] have been suggested to reduce performance loss.

Enhancing NoC Reliability: In NoC, both transient and permanent faults can manifest during transmission. Cyclic Redundancy Check (CRC) [19] is a basic transient fault detection technique 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 usually deployed. Single-bit error correction, double-bit error detection (SECDED) is one of the most commonly used ECC techniques in NoCs [20]. To handle permanent faults caused by transistor aging [21], a number of techniques have been proposed using load-balancing [22], circuitry redundancy [23], and adaptive routing techniques [20] among others. We should note that most of the techniques are static in nature where CRCs or SECDEDs are deployed all the time, regardless of if there are faults or not. Reliability enhancement mechanisms based on static techniques have been shown to require excessive power consumption, and longer delays, and thus significantly degrading NoC performance [23–25].

3 INTELLINOC ARCHITECTURE

In this section, we describe the micro-architecture and circuit-level details of the proposed IntelliNoC design. The overall IntelliNoC architecture is shown in Fig. 1. It consists of adaptive inter-router links based on MFACs, dynamic ECC hardware within each router, a stress-relaxing bypass route for power savings and stress management, and RL control. We describe the implementation of MFACs in Section 3.1, the adaptive ECC mechanism in Section 3.2, and the stress-relaxing technique in Section 3.3 below.

3.1 Multi-Function Adaptive Channels (MFACs)

Previous research [9, 10, 22] has shown that the excessive power consumption of router buffers can be reduced by moving storage to the

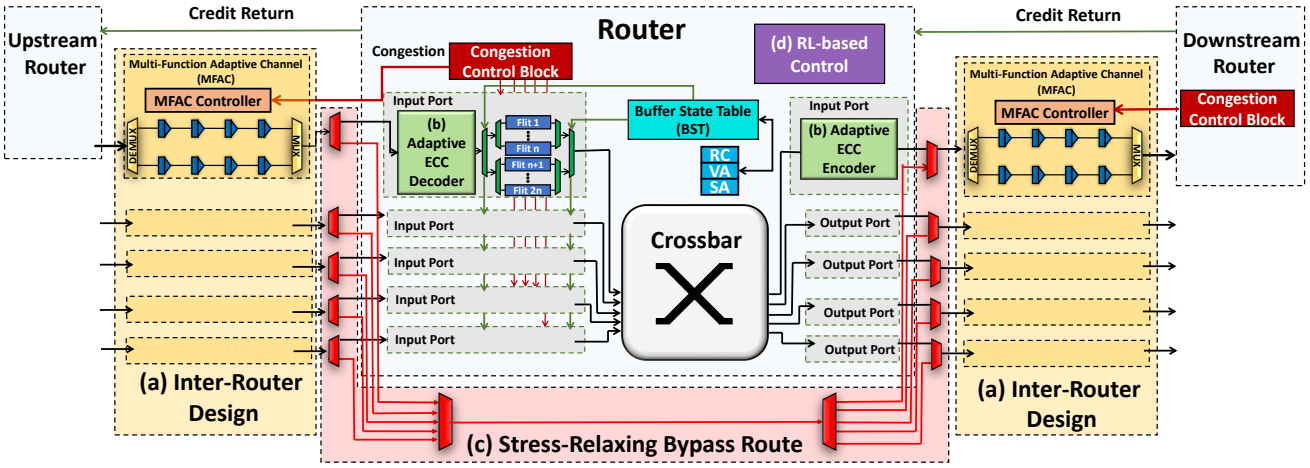

Fig. 1: IntelliNoC architecture design. IntelliNoC consists of (a) inter-router design using multi-function adaptive channel (MFAC) buffers, (b) per-router adaptive error detection/correction design, (c) stress-relaxing bypass route design, using (d) reinforcement learning (RL) based control policy.

inter-router links or channels. In this paper, we extend iDEAL channel buffers [10] to multi-function adaptive channel buffers (MFAC buffers), with the objective of reducing power, improving performance, and enhancing reliability. Much like with iDEAL buffers, we partially remove router buffers and use MFACs as storage and dynamically allocate link and router storage according to traffic patterns to reduce power consumption, with the side benefit of balancing wear-out. MFACs are different from iDEAL buffers in several ways. First, we evenly allocate channel buffers on two physical links in each channel to alleviate congestion and head-of-line blocking of the single link channel design, as shown in Fig. 2. Next, we use an MFAC controller to dynamically and independently configure the transmission/buffer functions of the physical links to perform multiple MFAC functions shown in Fig. 3. In doing so, MFACs are capable of four functions as opposed to two in iDEAL. As described in detail in Section 3.1.1, MFACs can function as (1) transmission repeaters, (2) link storage, (3) re-transmission buffers, and (4) relaxed timing buffers. The mode selection is explained in Section 4. Since our design may lead to a latency penalty (due to control overhead of the MFAC buffers) and potential congestion (due to head-of-line blocking of router buffers), we use dynamic router buffer allocation to maximize network throughput, as detailed in Section 3.1.2.

3.1.1 MFAC Buffers. Conventional channel buffers [10] use three-state transistors to propagate or store flits, controlled by a 1-bit flow control signal sent from the congestion control block [10]. Upon receiving a low congestion signal, the three-state repeaters are configured for transmission and propagate the flits. When the congestion signal is high, the tri-state transistors act as a storage device and buffer the flit. Fig. 2 shows the proposed MFAC buffers using the three-state transistors. The proposed MFAC buffers are comprised of two physical links with four buffer stages per link. Each physical link can be used either for storage or transmission (as regular repeaters). We added a new function-select controller, or the *MFAC controller* to the design. With the added control, the MFAC

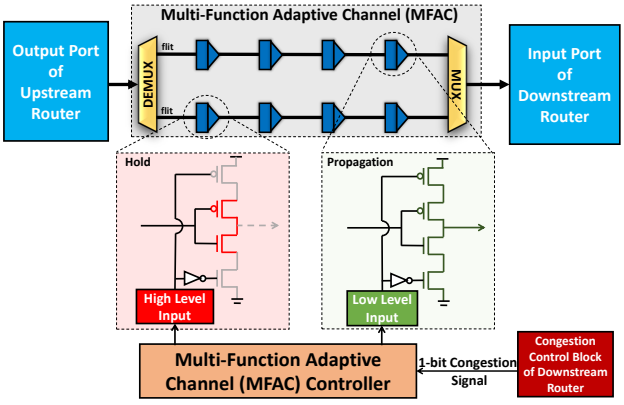

Fig. 2: Proposed multi-function adaptive channel (MFAC). Each MFAC is comprised of two physical links with four buffer stages per link.

buffers can now implement two additional functions: re-transmission buffers and relaxed timing buffers. The circuit details of the MFAC controller is shown in Fig. 3 and described below:

(1) Transmission Repeater (Fig. 3(a)): With this function, both MFAC buffer links are configured as repeaters. When the MFAC 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 MFAC to act as transmission channel.

(2) Link Storage (Fig. 3(b)): In this case, both MFAC buffer links are configured as link storage. When the MFAC 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 transistors' capacitance.

(3) Re-transmission Buffer (Fig. 3(c)): In this case, we use one of the MFAC buffer links to transmit flits, whereas the other MFAC

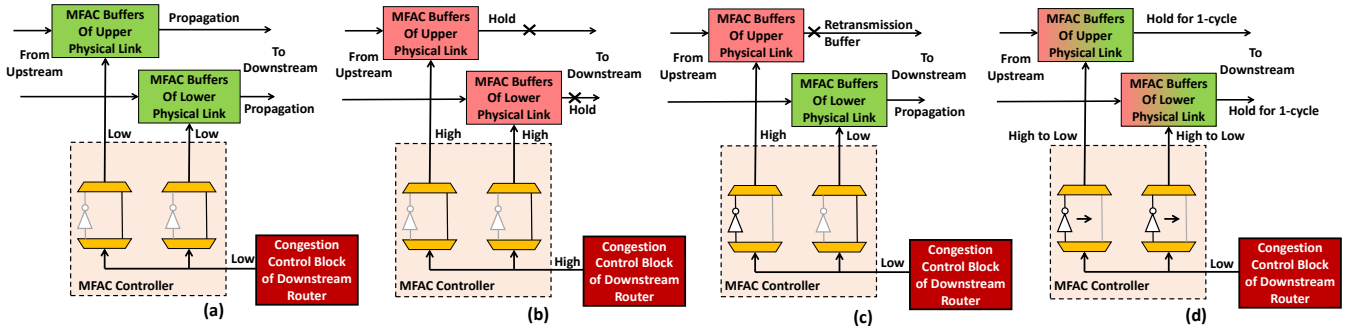

Fig. 3: Multi-function adaptive channel (MFAC) buffers assume four different functions: (a) regular repeaters for flit transmission, (b) regular buffers for storage on the link itself, (c) re-transmission buffers and (d) relaxed timing buffers for improved reliability.

buffer link is used to store flits for re-transmission purposes. In conventional SECDED design, a copy of the transmitted flit is stored in the local re-transmission buffer (in the upstream router) until it receives an acknowledgement (ACK) message back from the downstream router. The implementation of local re-transmission buffers can lead to excessive power and area overhead, especially since these re-transmission buffers are under-utilized when error levels are low. Therefore, it is beneficial to replace the traditional in-router re-transmission buffers with MFAC buffers, because the original flit will only be stored when needed (under higher error rates). Under this condition, the MFAC controller will send the same packet/flits on both MFAC buffer links. The MFAC controller configures the upper MFAC buffer link for storage (by applying a "hold" signal) and the lower MFAC buffer link for forwarding the flit (regular transmission). Upon receiving a NACK signal, the MFAC controller will release the flit for re-transmission. If ACK signal is received, the flit is discarded since the original transmission is error-free.

(4) Relaxed Timing Buffer (Fig. 3(d)): The proposed MFAC buffer links are able to function as relaxed timing transmission buffers, which can reduce the probability of transient errors by doubling the link traversal time [26]. When the MFAC controller receives a low signal (no congestion), it will reverse the signal from low to high to hold the flit for one cycle. After one clock-cycle delay, the MFAC controller will propagate the low signal to the MFAC buffers to propagate data. By storing for one additional cycle, we can provide a relaxed buffer design to reduce the probability of transient error.

3.1.2 Buffer Allocation and Flow Control. In conventional router design, each input port of the router is associated with a VC state table. The table records the state for each incoming flit and ensures that the head and the body flits are routed to the correct output port [27]. Dynamic buffer allocation schemes have been proposed to prevent performance degradation [11]. In such designs, the VC state table is extended to include the VC identifier (VC), read pointer (RP), write pointer (WP), allocated output port (OP), output VC (OVC), status (Stat), and credit count (CR). However, since each VC state table can only be accessed by its associated input port, such a design cannot be used when the router (and associated input ports) is powered off.

In IntelliNoC, we implement dynamic buffer allocation with a new 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. Moreover, the routing information in the BST is not powered off and can be accessed to route flits via bypass path when the router is powered off. Specifically, we add several new entries to the BST (which are shown in yellow and orange in Fig. 4) to retain VC information when the router is powered off. In the unified BST, the input port identifier (Port) indicates the input port of the incoming flit. The downstream router status (DRS) indicates if the downstream router associated to the current OP is power-gated or bypassed. The channel buffer pointer (CBP) and channel buffer credit (CBC) indicates the occupancy status of the associated MFAC buffers.

In conventional flow control, the header flit carries the packet information for route computation, VC allocation, and flow control. The output port and VC information are recorded in the VC table by the header flit and therefore body flits simply follow the VCID to find the correct output port from the VC table. The buffer allocation and flow control are similar to conventional design when the router is powered on. When the router is power-gated, the BST is still active and mimics the process of updating BST, as if the router is still switched on. Specifically, the BST 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 when the router is powered on or off, since all critical information is updated timely. It must be noted that while power-gated, the router will distribute the credits on MFAC buffers (CBC), since the credits of router buffers are unavailable. To enable BST functioning even when the router is power-gated, we consider a separate supply voltage that is not powered-off.

Additionally, the congestion control block monitors and updates the BST by recording all the available router buffer and MFAC buffer slots. If all the router buffer slots and MFAC buffer slots of an input direction are occupied, a congestion signal will be triggered. The proposed design is deadlock free, since we still maintain the virtual channels, which can avoid both protocol and routing deadlock. On the other hand, the head-of-line blocking in the channel buffers can be overcome by dynamic buffer allocation using the proposed unified BST table [11].

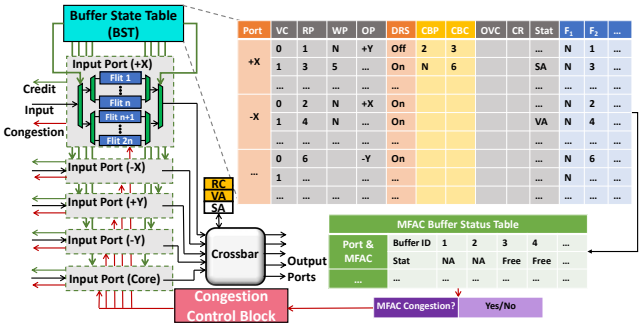

Fig. 4: Proposed unified buffer state table (BST). The green arrows indicate buffer slot allocation and credit signals by BST, while congestion signals are shown with red arrows.

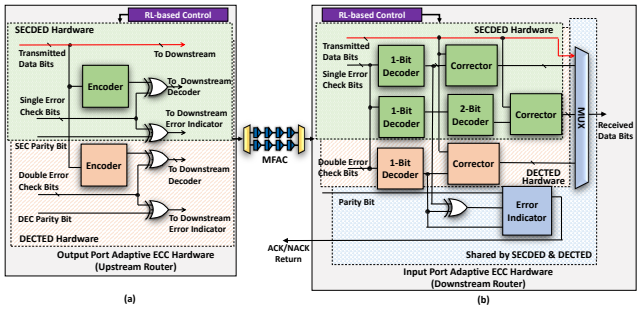

Fig. 5: Proposed adaptive error correction hardware. (a) Adaptive ECC encoding hardware located in upstream router's output port. (b) Adaptive ECC decoding hardware located in downstream router's input port. SECDED is active when logic circuits in green and blue are enabled, DECTED is active when logic circuits in green, orange, and blue are enabled. The red arrow shows flits with CRC enabled.

3.2 Adaptive Error Correction Hardware

Static error control schemes are either costly or not powerful enough to take advantage of traffic variations. Therefore, we propose adaptive error correction (or adaptive ECC) hardware, which can proactively and dynamically deploy the most suitable error correcting code (ECC) on a per-router basis. The different ECC coding includes end-to-end cyclic redundancy (CRC), per-hop SECDED (single-bit error correction, double-bit error detection) and per-hop DECTED (double-bit error correction, triple-bit error detection) [28, 29]. While CRC-only routers can solely detect errors at the destination, SECDED and DECTED can provide more powerful error detection and correction of the transmitted flits at each hop. For a SECDED/DECTED enabled router, an additional error detection encoder is assigned to each output port, and error detection decoders are added to input ports. When a router transmits flits to the downstream router, a copy of the transmitted flit is buffered in the current router's virtual channel (VC) until it receives an ACK message back from the downstream router. If a negative-acknowledgement (NACK) is received, the buffered flit will be re-transmitted to the downstream router. The circuit-level details of SECDED and DECTED encoders/decoders are shown in Fig. 5. In this paper, we propose a new adaptive error correction hardware which can act as DECTED hardware when it is fully activated, as SECDED when partially activated, or be entirely power-gated to only enable basic CRC, as shown in Fig. 5. The proposed dynamic ECC hardware functions as SECDED circuit when the combinational logic circuits in green and blue are enabled. Additionally, the proposed hardware functions as DECTED if all of the circuitry in green, blue, and orange are activated. The dynamic configuration of adaptive ECC hardware is guided by the reinforcement learning (RL) based control policy discussed in Section 5.

3.3 Stress-Relaxing Router Bypass

When the network experiences high traffic load, network resources, such as buffers, routers, and links, are highly utilized. This tends to raise their operating temperature and run-time stress, contributing to faults in the network. In this paper, we explore a stress-relaxing techniques where we proactively power-gate and bypass the NoC router to lower the temperature and reduce stress time, as shown in Fig. 6.

Conventional power-gating techniques reduce network connectivity (disconnect routers from the network) and incur large network latency, which negatively affects NoC performance. Several power-gating techniques [13, 14, 30] use a simple switch to route the low and sporadic traffic through the bypass links to keep network connectivity. When the router is power-gated, the incoming flits will be stored at the MFAC buffers located in the channel, and propagated using a round robin scheme. In this paper, we have adopted a similar idea, but deployed the new MFAC buffers to provide increased link storage. Unlike prior router bypass designs, the proposed bypass design allows us to extend the power-gated time and the bypass is operational for even low-to-moderate traffic load, since MFAC buffers provide extra storage and will not force the router to be powered-on. Further, to relax the stress-time, we will power-gate the router to mitigate wear-out and aging. When the router is powered-off, the incoming flits are stored at the MFAC buffers and forwarded by a simple round robin arbiter. As discussed in Section 3.1.2, the proposed design continues to utilize BST for routing information under power-gated conditions. In addition, we propose to use ML to predict power-gating opportunities to power off routers, therefore yielding optimal static power savings and reducing router stress-time.

4 PROACTIVE OPERATION MODES

In this section, we propose five proactive operation modes for IntelliNoC routers. Each operation mode has various MFAC configurations, error correction, re-transmission, and power management strategies. Occasionally, each IntelliNoC router independently selects and deploys an operation mode proactively, using a reinforcement learning based control policy (described in Section 5). The operation modes are fully detailed below.

- **Operation Mode 0 - Stress-Relaxing Mode:** In this mode, the stress-relaxing bypass route is activated, which means the entire router is power-gated, and the MFAC buffers are used to store the incoming flits. If the router is underutilized, or a high risk of

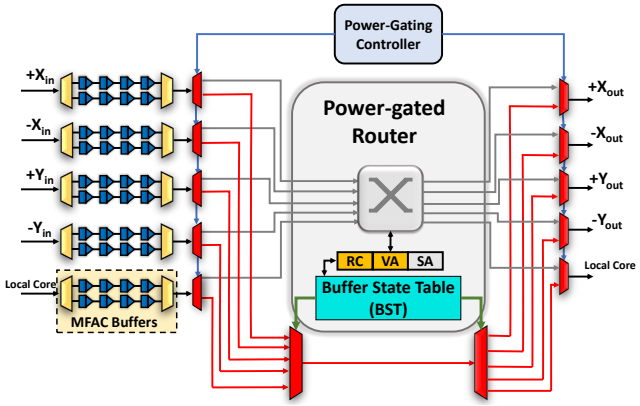

Fig. 6: Proposed stress-relaxing bypass route design. When the router is power-gated, the associated power-gating controller configures the MUX/DEMUX to enable the bypass route (shown in red). RC, VA, and BST are still active for route computing and MFAC buffer allocation.

overheating is predicted, this operation mode will be triggered. By reducing operating temperature and stress time, this mode can reduce the probability of the occurrence of permanent faults, along with significant static power savings.

- **Operation Mode 1 - Basic Error Detection Mode:** In this mode, the router disables the entire adaptive error correction hardware, and deploys basic CRC at the local core injection port for error detection. This operation mode is used under low traffic intensity, which often is associated with low error levels. To achieve maximum power savings and higher throughput, the MFAC buffer links are configured as storage buffers.
- **Operation Mode 2 - Per-hop SECDED Mode:** In this mode, the router switches its adaptive ECC hardware to SECDED to enable per-hop error detection and correction. This operation mode is beneficial when SECDED can handle most of the faults. Otherwise, it will either lead to unnecessary power and latency penalties (when error level is low) or excessive re-transmissions (when errors cannot be corrected by applying SECDED). The MFAC buffers are configured as re-transmission buffers.
- **Operation Mode 3 - Per-hop DECTED Mode:** In this mode, the router activates the entire adaptive ECC hardware to enable DECTED. This is the situation where the flits are more likely to contain errors in more than 1 bit position. The MFAC buffers are configured as re-transmission buffers.
- **Operation Mode 4 - Relaxed Transmission Mode:** In this mode, the errors in transmitted flits are beyond the error correction capability of SECDED/DECTED hardware. In this case, an additional clock cycle is inserted at every stage of the MFAC buffers for all flits. This mechanism doubles the link traversal time and relaxes the timing constraint on the flit transmission, so that the probability of timing errors can be reduced to near zero [26].

The dynamic selection of operation modes is performed by each router independently yet simultaneously, in a sequence of discrete time steps, using a reinforcement learning (RL) based control policy presented in Section 5. 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. By doing so, the downstream router will be informed to configure the ECC decoder located in the corresponding input port to apply the correct ECC coding, so 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 per-router error detection/correction and power-gating scheme provide a higher degree of freedom for individual routers to apply the best strategy at a given time,resulting in greater benefits for the entire network.

5 REINFORCEMENT LEARNING

In this section, we present a per-router reinforcement learning (RL) based control policy for dynamically selecting one of the IntelliNoC operation modes proposed above. RL refers to the training of machine learning models that optimizes the behavior of autonomous agents. In RL, an agent interacts with an environment and takes actions that can change the state of the environment with the aim of maximizing the total reward, or long-term return [15]. A NoC system can be seen as a cooperative multi-agent system, wherein the agents are components such as routers, links, or power systems linked to NoCs, and the environment is the entire communication system.

In RL, the *agent* (NoC router) acts as a learner and a decision maker and interacts with the *environment* (entire NoC system) in a sequence of discrete time steps. At each time step, the router observes the current *state s* by extracting runtime system metrics, selects an *action a* from one of the proposed operation modes, and applies it at the next step. At the following time step, upon taking the action, the NoC metrics change and result in a new state s', which is fed back to the agent. In addition to observing the new state, the agent also receives a *reward r* (a function of energy, performance, and reliability), representing the impact of the action on system performance. A *policy* π maps states and actions, specifying how to choose actions given the state of the environment. The goal of the RL algorithm is to learn a policy that maximizes the agent's long-term *return* $\mathscr{R}_t$, calculated as the exponentially *discounted* sum of future rewards: $\mathscr{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$ [14]. In this paper, the Q-learning algorithm [15] is used to learn the optimal policy by estimating an *action-value function* $Q(s,a)$. It represents the maximum return that the agent is expected to receive if it starts in state s, takes action a, and follows the optimal policy for the remaining actions. If $Q(s,a)$ is the optimal action-value function, then the corresponding optimal policy is $\pi(s) = \arg\max_a Q(s,a)$.

State Space and Action Space: The NoC system metrics, or *features*, comprising the RL state are listed in Fig. 7. These metrics are local to each RL agent and are monitored at each time step. The action space $A = \{a_0, a_1, a_2, a_3, a_4\}$ contains the five operation modes that the RL agents can select from, as described in Section 4.

Reward Function: The router agents have the holistic goal of minimizing network latency, reducing power consumption, and decelerating the aging of the NoC system. Therefore, the reward function for router i at time step t is defined as:

$$r_{i,t} = -\log\left(Latency_{i,t}\right) - \log\left(Power_{i,t}\right) - \log\left(Aging_{i,t}\right) \quad (1)$$

The *Latency* in the equation above refers to the average end-to-end latency of the specific router i, which can be obtained by calculating

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
Other Metrics	16. Temperature	Local router temperature in ℃

Fig. 7: Features in the *state* vector of each router.

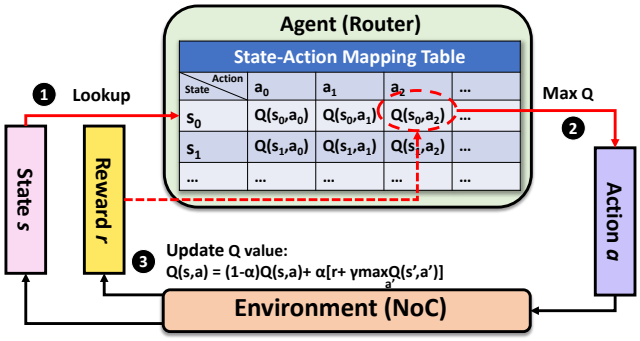

Fig. 8: Q-learning process. At time step t, the action a_2 with maximum Q-value for current state s_0 is selected. The reward for action a_2 will be calculated after a_2 impacts NoC environment. Then the Q-value will be updated following the temporal difference rule (2).

the difference between the flit injection time and the ACK message injection time for each flit transmission within time step t. ACK message injection time represents the time when the flit arrives at the destination node and is accepted. Additionally, the average power consumption is monitored by NoC sensors (power modules) in the next time step. It contains both static power consumption and dynamic power consumption. The aging factor is calculated using the aging model, which is described in detail in Section 6.2. The latency, power, and aging variables used in Equation 1 are always larger than 1, thus precluding the reward from taking extremely large values. Working in log-space also makes any difference in scale among the three quantities immaterial, as multiplying them with different constant factors translates into one constant term in the reward and Q-values, without any impact on the Q-learning algorithm.

Q-Learning: To find the optimal action-value function *Q(s, a)* that maximizes the expected return, we use the tabular Q-learning algorithm [15]. Assuming the state s is discrete, a table Q is initialized with zeros for all possible (s,a) pairs. 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. This is achieved by using an ε-greedy policy which takes a random action with a small probability ε, and the maximum value action with probability $1-\varepsilon$. After taking an action a and observing the reward r and new state s', the action-value table entry *Q(s, a)* is updated using the following temporal difference rule:

$$Q(s,a) = (1-\alpha)\,Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right] \qquad (2)$$

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 action-value function and its corresponding optimal policy [15]. The variable γ (where $0 \leq \gamma \leq 1$) in this equation is the *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. Additionally, an ε-greedy policy is also applied to explore unvisited regions of the state-action space [15, 31, 32]. A detailed discussion of how the parameters γ and ε impact system-level performance is presented in Section 7.3.

The Q-values for all possible state-action pairs are recorded independently in a local state-action mapping table for each router. Thus, the computational overhead of RL is determined by the traversal latency of the state-action mapping table. Some of the features selected in Fig. 7 have continuous values (*e.g.* temperature), as such they need to be discretized. In this paper, the feature values are evenly discretized into five bins according to the range of each feature through benchmark profiling.

Fig. 8 demonstrates the working of the RL-based control logic when running a benchmark. At each time step, the process goes through several stages. In stage ❶, the router uses the state s to look up the local state-action mapping table for a matching row (in Fig. 8, we assume current state s matches state s_0 in the mapping table). In stage ❷, the router selects an action a (one of five operation modes), which has the maximum $Q(s,a)$-value among all possible actions for state s, for the next time step (we assume a_2 in Fig. 8 has the maximum Q-value). Upon taking the action a, the NoC system transits to a new state s'. In stage ❸, the NoC system provides a reward r (defined in Equation 1) to the router. The reward will be used in the temporal difference rule shown in Equation 2 to update $Q(s,a)$. Each router will go through the three stages at each time step. The initialization of the per-router controller and the state-action mapping table will be discussed in Section 6.3.

6 EVALUATION METHODOLOGY

In this section, we present the fault injection model for transient faults, the transistor aging model for permanent faults, and the simulation framework. In this work, we only consider error transmission caused in the inter-router links. In future work, we will consider faults in the control circuit, routing table, state-action table, and other sources.

6.1 Fault Model for Transient Fault Injection

To assess the reliability improvements of the proposed IntelliNoC, we introduce a fault model to realistically produce a probability of timing errors occurring for each link using a combination of error models and simulators. The simulators and error models include

VARIUS [33] fault model, NoC fault model [34] and HotSpot [35] thermal model. These models and simulators are fully modified and combined with the network simulator, so that transient faults can be injected dynamically at runtime. Per-router RL agents monitor and predict the supply voltage, operation frequency, and link utilization. These values are injected in HotSpot to obtain router operating temperature at runtime. The temperature values (generated by HOTSPOT) are fed into the VARIUS timing error model to generate probability of timing errors (R_e) for each transmitted bit. This bit error rate R_e increases as operating temperature increases or as supply voltage decreases. Using R_e, we can calculate the error rate of a transmission flit (P_{fault}). Based on the above, we calculate the probability of a faulty n-bit flit as follows:

$$P_{fault} = 1 - \left(1 - R_e\right)^n \tag{3}$$

6.2 Modeling Aging in NoCs

We model and calculate the *aging* factor in (1) by correlating the shift in the threshold voltage of the transistor (ΔV_{th}). ΔV_{th} shift is tied to the wear-out effect of the transistors due to long-term stress according to the Alpha Power Law [36]:

$$d_g \propto \frac{V_{dd}}{\mu \left(V_{dd} - V_{th}\right)^{\alpha}} \tag{4}$$

The equation above indicates that a shift in the threshold voltage V_{th} leads to a variation in circuit delay d_g. For a single transistor, when ΔV_{th} reaches a certain level, the delay degradation will exceed the margins within which the transistor operates correctly. Since the transistor cannot sample correctly, the circuit will fail and is considered to be a permanent fault. We consider a permanent fault in the transistor when ΔV_{th} is greater than 10% [37].

Typically, Negative Bias Temperature Instability (NBTI) [38, 39] and Hot Carrier Injection (HCI) [21, 40] contribute to shift in ΔV_{th}. Research [21] has shown that the shift in threshold voltage caused by NBTI and HCI are independent and correspond to the stress of p-type metal-oxide semiconductor (PMOS) and n-type metal-oxide semiconductor (NMOS) transistors respectively. Therefore, we use both NBTI- and HCI-caused ΔV_{th} to quantify the *aging* factor.

Specifically, ΔV_{th_NBTI} is given by [38, 39]:

$$\Delta V_{th\_NBTI} = A\left(\left(1+\delta\right)t_{ox} + \sqrt{C\left(t - t_0\right)}\right)^{2n} \tag{5}$$

From (5), ΔV_{th_NBTI} is correlated to A which has an exponential dependence on operating temperature and time $t - t_0$, whereas δ, n, t_{ox}, and C are all device related constants [37]. We monitor the operating temperature and time to calculate ΔV_{th_NBTI}.

On the other hand, ΔV_{th_HCI} is given by equation 6 below [21, 40]:

$$\begin{aligned} \Delta V_{th\_HCI} &= A_{HCI} \cdot I^m \cdot t_{stress}^n, \\ where\ \ t_{stress} &= d_{g0} \cdot f \cdot \alpha_{SA} \cdot t_{runtime} \end{aligned} \tag{6}$$

where A_{HCI} and I are material-dependent parameters, m and n are technology-related exponents, and α_{SA} is switching activity. All of them are set to default values [37]. d_{g0} (the transition delay) and f (clock frequency) are captured by the simulator.

Using NBTI- and HCI- induced shift in threshold voltage, we model the *Aging* factor given in equation (1) as follows:

Table 1: Simulation Environment Setup

# of cores	64 out-of-order CPUs @ 32nm
Voltage and Frequency	1.0 Volt, 2.0 GHz
NoC Parameters	8 × 8 2D Mesh, X-Y routing, 4-stage routers
Packet Size	4 × 128-bit flits
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) 8CB × 2 sub-networks (EB) 2RB-4VC-8CB (CP and CPD) 2RB-4VC-8CB (IntelliNoC)

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

$$\begin{cases} \Delta V_{th} = \Delta V_{th\_NBTI} + \Delta V_{th\_HCI} \\ Aging = 1 + \dfrac{\Delta V_{th}}{V_{th0}} \times 100\% \end{cases} \tag{7}$$

It must be noted that the *Aging* factor is designed to have a value greater than 1 so that it can be used in the reward function (1), as discussed in Section 5.

6.3 Simulation Setup

We evaluate our proposed design using a modified version of the cycle-accurate network simulator *Booksim2* [17], where we fully incorporate the fault models and RL techniques. We also use Netrace [41] to capture cycle-accurate benchmark traces for the network simulator. Table 1 describes the simulation parameters used. The selection of RL parameters (such as α, γ, and ε) can impact the performance of the trained control policy [31, 32, 42]. We tune the discount rate γ and exploration probability ε on blackscholes benchmark from PARSEC, resulting in $\gamma = 0.9$ and $\varepsilon = 0.05$. The learning rate α is set to the default value of 0.1. A more detailed discussion on the tuning process is provided in Section 7.3. The operation modes of all routers are initialized to mode 1.

Workloads from the PARSEC benchmark suite [43] are tested. Benchmarks from PARSEC are transformed into a trace file by the Netrace simulator. These trace files contain packet injection/ejection events and offer runtime information (such as time, packet size, transmission source, destination, or event type). We compare the performance of the IntelliNoC design to the baseline which consists of a traditional wormhole-based router and static SECDED hardware. We also compare our IntelliNoC design with several other state-of-the-art techniques including, Elastic Buffers (EB) [9], iDEAL channel buffers [10] with power gating (CP), and extended CP with dynamic ECC capabilities (CPD). For CPD, at each time step, the selection of ECC hardware is based on the error level of the previous time step. The agent calculates which error type is most common (no errors in a flit, 1-bit error per flit, 2-bit errors per flit, or more than 3-bit errors per flit). For the RL-based IntelliNoC, we pre-train the per-router policy using blackscholes, the same benchmark that was used for tuning (pre-training on other benchmarks led to similar performance). After that, we use the other applications in PARSEC

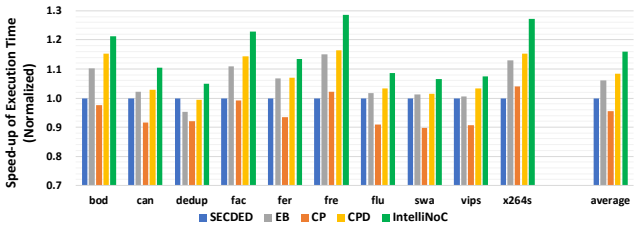

Fig. 9: Speed-up of full application execution time comparison, normalized to the SECDED baseline (higher is better).

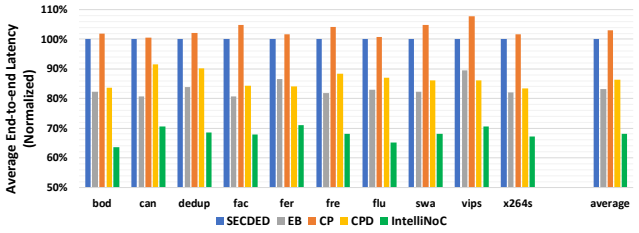

Fig. 10: Average end-to-end latency comparison, normalized to the SECDED baseline (lower is better).

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 (2) every 1000 cycles.

7 RESULTS AND ANALYSIS

7.1 Performance Analysis

Speed-up: The speed-up is obtained by calculating the ratio of the full application execution time of various techniques (SECDED, EB, CP, and CPD) to the execution time using the proposed IntelliNoC running various benchmarks, as shown in Fig. 9. As can be seen in Fig. 9, IntelliNoC has the largest speed-up over all techniques evaluated. EB achieves 6% speed-up over the SECDED baseline, since it shortens the router pipeline stages by eliminating VA stage. CP results in 3% performance loss because of degraded throughput and the power-gating wake-up latency. Adaptive error control (in CPD and IntelliNoC) successfully accelerates benchmark execution (by 8% and 16% respectively) by reducing ECC overhead (via CRC) and re-transmission traffic (as discussed in Section 7.2).

Average End-to-End Latency: Fig. 10 shows the normalized end-to-end packet latency for different techniques. It can be seen that the proposed IntelliNoC framework achieves an average of 32% end-to-end latency reduction. It should be noted that EB achieves 17% end-to-end latency reduction over the baseline due to the elimination of VA stage in router pipeline. However, re-transmission traffic can be excessive in traditional routers with static error correction techniques. IntelliNoC applies appropriate ECC mode and relaxing strategies to minimize transient and permanent faults, which in turn reduces the re-transmission traffic.

Overall Static Power Consumption: Fig. 11 shows overall static power consumption for the various techniques. EB, with a zero-router-buffer design, reduces static power consumption by 14%,

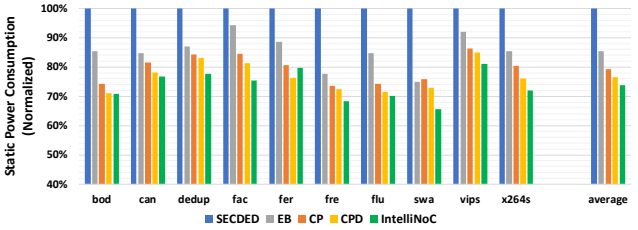

Fig. 11: Overall static power consumption comparison, normalized to the SECDED baseline (lower is better).

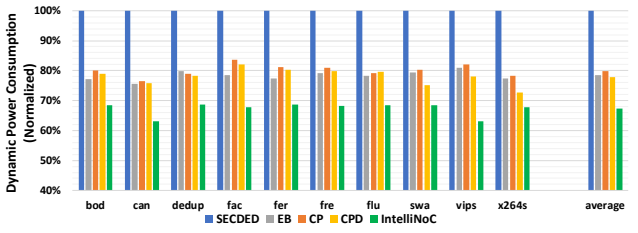

Fig. 12: Overall dynamic power consumption comparison, normalized to the SECDED baseline (lower is better).

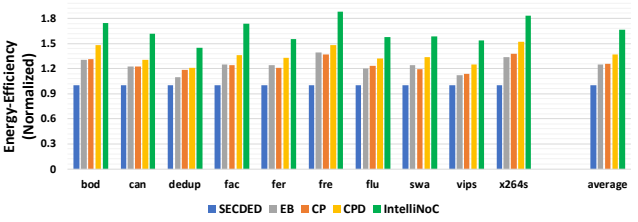

Fig. 13: Energy-efficiency comparison, normalized to the SECDED baseline (higher is better).

while CP achieves 20% static power savings due to power-gating. The use of adaptive error control in CPD provides 23% static power savings on an average, since reducing the re-transmission messages provides more opportunities for power-gating the router. However, in some benchmarks (*e.g.* ferret), CPD performance is worse than CP, due to the control policy. IntelliNoC provides maximum static power savings for all applications due to the dynamic nature of the RL-based control policy and the better choices made for the operating modes.

Overall Dynamic Power Consumption: Dynamic power reduction is achieved by reducing the number of router buffers (EB, CP) and/or by mitigating faults and reducing the number of re-transmissions. IntelliNoC, using MFACs and dynamic error control, is able to reduce re-transmission traffic significantly. As a result, IntelliNoC outperforms all other techniques in reducing dynamic power consumption as shown in Fig. 12.

Energy-Efficiency: We define energy-efficiency as:

$$Energy\text{-}Efficiency = \left[\left(P_{static} + P_{dynamic}\right) \times T_{exec}\right]^{-1} \quad (8)$$

P_{static} and $P_{dynamic}$ are static and dynamic power consumption, and T_{exec} is the benchmark execution time, which are obtained through

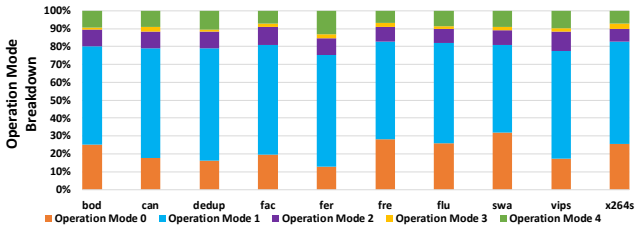

Fig. 14: Operation mode breakdown.

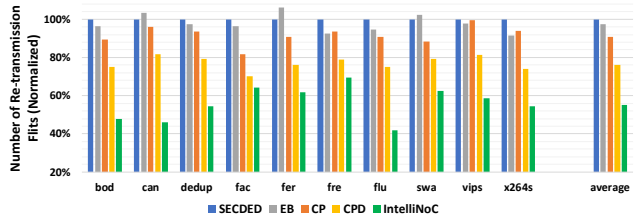

Fig. 15: Number of re-transmission flits comparison, normalized to the SECDED baseline (lower is better).

Booksim2. Fig. 13 shows the energy-efficiency measurements for all techniques studied and normalized to the SECDED baseline. IntelliNoC improves energy-efficiency by 67%, compared to the SECDED baseline, while the maximum energy-efficiency improvement using other techniques is 36% (CPD).

Operation Mode Breakdown: Fig. 14 shows the breakdown of operation modes in IntelliNoC architecture for all PARSEC benchmarks. We study the ratio of the number of clock cycles utilized by each operation mode to the total execution time. Mode 0 occupies 20% of the total execution time on average, leading to static power savings. This indicates that the stress-relaxing bypass route and the router is power-gated for 20% of the time. Basic CRC is sufficient 55% of the time (mode 1), which indicates low transient errors for half the execution of all applications. For the remaining 25% of the time, more powerful ECCs (SECDED, DECTED and relaxed transmission) are essential (basically modes 2 to 4) to reduce re-transmission traffic while still providing fault-tolerance coverage. By applying the RL-based control policy to balance performance, power consumption, and reliability, IntelliNoC dynamically selects the appropriate operation mode.

7.2 Reliability Analysis

Improvement in Transient Fault Tolerance: To examine the reliability improvement of IntelliNoC with respect to soft errors, we compare in Fig. 15 the amount of re-transmission traffic for four techniques: SECDED, EB, CP, and CPD, with that of IntelliNoC. Results are normalized to the SECDED baseline. As shown, all techniques are able to reduce the number of re-transmissions due to the fact that they all reduce router buffers. This results in reduced operating temperatures, in turn reducing timing errors. However, IntelliNoC, with its ability to choose optimized control policy and more powerful error correcting schemes, achieves the largest reduction on re-transmissions at 45%.

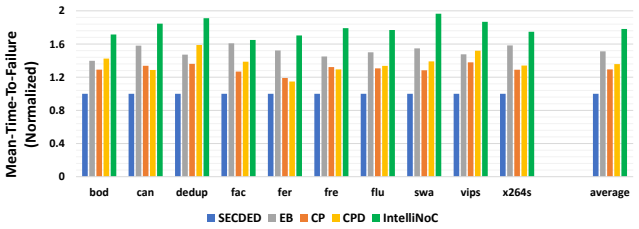

Fig. 16: Mean-time-to-failure (MTTF) comparison, normalized to the SECDED baseline (higher is better).

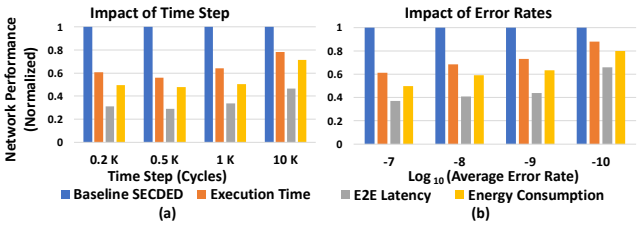

Fig. 17: Impact of (a) RL time step and (b) transient error rates on network performance metrics. Results are normalized to the SECDED baseline.

Improvement in Permanent Fault Tolerance: To quantify the reliability improvement of the proposed framework with respect to permanent faults, we use the mean-time-to-failure (MTTF) metric. To estimate MTTF, we calculate Failure-in-Time (FIT) of the baseline circuitry and the correction circuitry using the architectural level reliability-modeling framework proposed in [23, 44]. We then use the permanent fault model discussed in Section 6.2 and use the FIT value above to calculate material related parameters. A normalized MTTF that compares IntelliNoC with other designs is shown in Fig. 16. As can be seen in Fig. 16, the MTTF of IntelliNoC is 1.77 times the MTTF of the baseline, indicating that the proposed design is 77% more reliable than the baseline. Although EB, CP, and CPD also achieve improved MTTF values, simulation results still indicate that the stress-relaxing feature of IntelliNoC plays an important role in further improving the robustness of the system.

7.3 Sensitivity Analysis

Impact of Different Time Steps: In order to study the impact of the RL time step (number of clock cycles of each time step), we varied the time step t starting from 200 to 10,000 clock cycles. We evaluated three NoC performance metrics: execution time, end-to-end packet latency, and energy consumption of full benchmark suite, and compared the results to the SECDED baseline. The evaluation results are illustrated in Fig. 17(a). As can be seen in Fig. 17(a), a longer time step (10K cycles) can result in sub-optimal system-level performance. Longer cycle time (10K cycles) hurts performance since the RL agent selects the mode according to the average value for the entire time step of 10K cycles. On the other hand, aggressively reducing the length of time steps (200 clock cycles) also leads to a degradation of RL performance, since the computational overhead of RL dominates when compared to system-level performance.

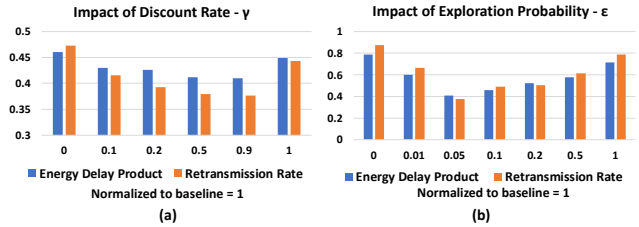

Fig. 18: Impact of (a) discount rate γ and (b) exploration probability ε of RL on network performance metrics. Results are normalized to the SECDED baseline.

Impact of Different Error Rates: In order to study the impact of different error rates on the proposed methodology, we artificially inject transient errors into the NoC system with average bit error rates equal to 10^{-7}, 10^{-8}, 10^{-9}, and 10^{-10}. Three NoC performance metrics are evaluated: execution time, end-to-end latency, and energy consumption. The evaluation results are illustrated in the plot in Fig. 17(b). As can be seen in Fig. 17(b), the proposed design achieves better performance as the error rate increases.

Impact of Discount Rate γ : We discuss the impact of the RL parameter, discount rate γ on the energy-delay product (EDP). Lower EDP indicates better performance. The blackscholes of PARSEC benchmark is used for parameter tuning. As discussed in Section 5, the parameter γ controls how much importance the RL agent gives to future rewards. Fig. 18(a) shows that the system EDP improves initially with larger γ. However, 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 to 0.9.

Impact of Exploration Probability ε : Fig. 18(b) shows the impact of ε values on the system EDP. As ε increases from 0 to 1, the RL agent is more likely to take random actions, and thus explores state-action pairs that have yet to be tested. In the extreme case when ε is 0, the router agent selects the initial mode most of the time. Conversely, when ε is 1, the agent takes actions entirely at random. Both cases result in sub-optimal system performance. As shown in Fig. 18, the best energy-delay product is achieved when ε equals to 0.05.

7.4 Overhead Analysis

We evaluate the area overhead of IntelliNoC and other designs with Synopsys Design Vision software in 32nm 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 can be seen from the Table, CP and IntelliNoC require less area than the baseline because they both lower the number of router buffers and exploit channel buffers for storage. EB eliminates router buffers all together and therefore requires the least area (-32.7%).

In IntelliNoC, the use of RL incurs additional overheads. The overheads include (a) the computational and energy overheads from calculating Q-values and traversing Q-table at each time step and (b) the area overhead induced by Q-table storage. We use [45] to calculate the energy overhead of RL in IntelliNoC. Result shows that at each 1k cycle time step, the RL consumes 0.16 pJ, and its total timing overhead is estimated to be 5 cycles, which is negligible. We also use Synopsys to calculate the area overhead of Q-table. We design our state space using 16 features, each of which has been discretized into 5 bins, as shown in Fig. 7. However, during the pre-training phase using blackscholes benchmark, we observe that the Q-table is indeed small in size, with no greater than 300 entries. That may be because the selected features are correlated to each other, and some combinations of the feature values would never be achieved in benchmark execution. To ensure a sufficient storage for Q-values during test phase, we assign a Q-table with 350 entries to each router, which leads to an area overhead that consumes 4% of total router area.

Table 2: Area Overhead Comparison (μm^2)*.

	Baseline	EB	CP	IntelliNoC
Router Buffer	1248.3 ×16 /port	-	1248.3 ×8 /port	1248.3 ×8 /port
Crossbar	9004.7	11774.6	9004.7	9004.7
Channel	136.7	5790.4	2734.4	2869.6
ECC	3325.4	3325.4	3325.4	3940.3
Total	119807.0	80612.6	83953.1	89313.7
%Change	-	-32.7%	-29.9%	-25.4%

* The configurations of router buffer numbers, virtual channel numbers, and channel buffer numbers for different designs are shown in Table 1. The baseline uses the static SECDED baseline router. The area overhead of elastic buffers and channel buffers are included in "Channel" in Table 2.

8 CONCLUSIONS

In this paper, we propose **IntelliNoC**, an intelligent NoC design which can simultaneously achieve improved performance, energy-efficiency, and reliability using architectural innovations and RL for dynamic control. The new NoC design consists of multi-function adaptive inter-router channel buffers, per-router dynamic error correction hardware, stress-relaxing bypass design, five proactive operating modes, and a RL-based dynamic control policy. Each RL agent (*i.e.* router) learns from the NoC behavior and updates a control policy to select an optimal operating mode at any given time with the objective of achieving high performance, increased reliability, and reduced power consumption. Simulation using the PARSEC benchmark suite shows that the proposed **IntelliNoC** framework improves energy-efficiency by 67%, enhances mean-time-to-failure (MTTF) by 77%, decreases end-to-end packet latency by 32%, and lowers area requirements by 25% over baseline NoC architecture. These results demonstrate the amplifying and synergistic effects of integrating architectural innovations with machine learning in a holistic approach to substantial improvements in the network performance, energy-efficiency, and reliability for NoC designs.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *computer*, 35(1):70–78, 2002.

[2] William J. Dally and Brian Towles. Route packets, not wires: On-chip inteconnection networks. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 684–689. ACM, 2001.

[3] Konstantinos Aisopos, Andrew DeOrio, Li-Shiuan Peh, and Valeria Bertacco. Ariadne: Agnostic reconfiguration in a disconnected network environment. In *2011 International Conference on Parallel Architectures and Compilation Techniques*, PACT'11, pages 298–309, 2011.

[4] Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Vijaykrishnan Narayanan, Mazin S Yousif, and Chita R Das. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. *ACM SIGARCH Computer Architecture News*, 34(2):4–15, 2006.

[5] Ke Wang, Ahmed Louri, Avinash Karanth, and Razvan Bunescu. High-performance, energy-efficient, and fault-tolerant network-on-chip design using reinforcement learning. In *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, DATE'19, 2019.

[6] Pavan Poluri and Ahmed Louri. An improved router design for reliable on-chip networks. In *Proceedings of 28th International Parallel and Distributed Processing Symposium*, IPDPS'14, pages 283–292, 2014.

[7] Hiroki Matsutani, Michihiro Koibuchi, Daisuke Ikebuchi, Kimiyoshi Usami, Hiroshi Nakamura, and Hideharu Amano. Ultra fine-grained run-time power gating of on-chip routers for CMPs. In *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 61–68. IEEE, 2010.

[8] Mark Clark, Avinash Kodi, Razvan Bunescu, and Ahmed Louri. LEAD: learning-enabled energy-aware dynamic voltage/frequency scaling in NoCs. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, pages 82:1–82:6, 2018.

[9] George Michelogiannakis and William J Dally. Elastic buffer flow control for on-chip networks. *IEEE Transactions on Computers*, 62(2):295–309, 2013.

[10] Avinash Karanth Kodi, Ashwini Sarathy, and Ahmed Louri. iDEAL: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures. *ACM SIGARCH Computer Architecture News*, 36(3):241–250, 2008.

[11] Chrysostomos A Nicopoulos, Dongkook Park, Jongman Kim, Narayanan Vijaykrishnan, Mazin S Yousif, and Chita R Das. ViChaR: A dynamic virtual channel regulator for network-on-chip routers. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 333–346. IEEE Computer Society, 2006.

[12] Reetuparna Das, Satish Narayanasamy, Sudhir K Satpathy, and Ronald G Dreslinski. Catnap: Energy proportional multiple network-on-chip. *ACM SIGARCH Computer Architecture News*, 41(3):320–331, 2013.

[13] Ahmad Samih, Ren Wang, Anil Krishna, Christian Maciocco, Charlie Tai, and Yan Solihin. Energy-efficient interconnect via router parking. In *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, HPCA '13, pages 508–519, Washington, DC, USA, 2013. IEEE Computer Society.

[14] Hao Zheng and Ahmed Louri. An energy-efficient network-on-chip design using reinforcement learning. In *56th Design Automation Conference (DAC)*, 2019.

[15] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[16] Lucian Busoniu, Robert Babuska, and Bart De Schutter. Multi-agent reinforcement learning: A survey. In *Proceedings of 9th International Conference onControl, Automation, Robotics and Vision*, ICARCV'06, pages 1–6, 2006.

[17] Nan Jiang, James Balfour, Daniel U Becker, Brian Towles, William J Dally, George Michelogiannakis, and John Kim. A detailed and flexible cycle-accurate network-on-chip simulator. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software*, ISPASS'13, pages 86–96, 2013.

[18] Lizhong Chen and Timothy M Pinkston. Nord: Node-router decoupling for effective power-gating of on-chip routers. In *Intl. Symp. on Microarchitecture (MICRO)*, pages 270–281. IEEE Computer Society, 2012.

[19] Shu Lin and Daniel J Costello. *Error control coding*, volume 2. Prentice Hall Englewood Cliffs, 2004.

[20] David Fick, Andrew DeOrio, Jin Hu, Valeria Bertacco, David Blaauw, and Dennis Sylvester. Vicis: A reliable network for unreliable silicon. In *Proceedings of 46th ACM/EDAC/IEEE Annual Design Automation Conference*, DAC'09, pages 812–817, 2009.

[21] Dominik Lorenz, Georg Georgakos, and Ulf Schlichtmann. Aging analysis of circuit timing considering NBTIand HCI. In *Proceedings of the 15th IEEE International On-Line Testing Symposium, IOLTS'09*, pages 3–8. IEEE, 2009.

[22] Dominic DiTomaso, Avinash Kodi, and Ahmed Louri. QORE: A fault tolerant network-on-chip architecture with power-efficient quad-function channel (qfc) buffers. In *Proceedings of 20th International Symposium on High Performance Computer Architecture*, HPCA'14, pages 320–331, 2014.

[23] Kypros Constantinides, Stephen Plaza, Jason Blome, Bin Zhang, Valeria Bertacco, Scott Mahlke, Todd Austin, and Michael Orshansky. Bulletproof: A defect-tolerant cmp switch architecture. In *Proceedings of The 12th International Symposium on High-Performance Computer Architecture*, HPCA'06, pages 5–16, 2006.

[24] Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, and Timothy Mark Pinkston. A lightweight fault-tolerant mechanism for network-on-chip. In *Proceedings of 2nd ACM/IEEE International Symposium on Networks-on-Chip*, NOCS'08, pages 13–22, 2008.

[25] Yuechen Chen, Md Farhadur Reza, and Ahmed Louri. DEC-NoC: an approximate framework based on dynamic error control with applications to energy-efficient NoCs. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 480–487, Oct 2018.

[26] Dominic DiTomaso, Travis Boraten, Avinash Kodi, and Ahmed Louri. Dynamic error mitigation in NoCs using intelligent prediction techniques. In *Proceedings of 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO'16, pages 1–12.

[27] William J Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed systems*, 3(2):194–205, 1992.

[28] Jangwoo Kim, Nikos Hardavellas, Ken Mai, Babak Falsafi, and James Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the 40th annual IEEE/ACM international symposium on microarchitecture (MICRO)*, pages 197–209. IEEE Computer Society, 2007.

[29] Xiaowen Chen, Zhonghai Lu, Yuanwu Lei, Yaohua Wang, and Shenggang Chen. Multi-bit transient fault control for NoC links using 2D fault coding method. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8. IEEE, 2016.

[30] Hao Zheng and Ahmed Louri. Ez-pass: An energy & performance-efficient power-gating router architecture for scalable NoCs. *IEEE Computer Architecture Letters*, 17(1):88–91, 2018.

[31] Yuxin Bai, Victor W Lee, and Engin Ipek. Voltage regulator efficiency aware power management. In *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS'17, pages 825–838, 2017.

[32] Quintin Fettes, Mark Clark, Razvan Bunescu, Avinash Karanth, and Ahmed Louri. Dynamic voltage and frequency scaling in NoCs with supervised and reinforcement learning techniques. *IEEE Transactions on Computers*, 2018.

[33] Smruti R Sarangi, Brian Greskamp, Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. VARIUS: A model of process variation and resulting timing errors for microarchitects. *IEEE Transactions on Semiconductor Manufacturing*, 21(1):3–13, 2008.

[34] Andrew B Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, DATE'09, pages 423–428, 2009.

[35] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. HotSpot: A compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, 2006.

[36] Takayasu Sakurai and A Richard Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of solid-state circuits*, 25(2):584–594, 1990.

[37] Yao Wang, Sorin Cotofana, and Liang Fang. A unified aging model of NBTI and HCI degradation towards lifetime reliability management for nanoscale mosfet circuits. In *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 175–180. IEEE Computer Society, 2011.

[38] Muhammad Ashraful Alam and Souvik Mahapatra. A comprehensive model of PMOS NBTI degradation. *Microelectronics Reliability*, 45(1):71–81, 2005.

[39] Sarvesh Bhardwaj, Wenping Wang, Rakesh Vattikonda, Yu Cao, and Sarma Vrudhula. Predictive modeling of the NBTI effect for reliable design. In *Proceedings of Custom Integrated Circuits Conference*, CICC'06, pages 189–192. IEEE, 2006.

[40] Hyungjun Kim, Arseniy Vitkovskiy, Paul V Gratz, and Vassos Soteriou. Use it or lose it: Wear-out and lifetime in future chip multiprocessors. In *Intl. Symp. on Microarchitecture (MICRO)*, MICRO'13, pages 136–147, 2013.

[41] Joel Hestness, Boris Grot, and Stephen W Keckler. Netrace: dependency-driven trace-based network-on-chip simulation. In *Proceedings of the 3rd International Workshop on Network on Chip Architectures*, NoCArc'10, pages 31–36, 2010.

[42] Engin Ipek, Onur Mutlu, José F Martínez, and Rich Caruana. Self-optimizing memory controllers: A reinforcement learning approach. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 39–50, 2008.

[43] Christian Bienia and Kai Li. PARSEC 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[44] Jeonghee Shin, Victor Zyuban, Zhigang Hu, Jude A Rivers, and Pradip Bose. A framework for architecture-level lifetime reliability modeling. In *Proceedings of 37th Annual IEEE/IFIP International Conference onDependable Systems and Networks*, DSN'07, pages 534–543. IEEE, 2007.

[45] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *Proceedings of 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.