SecureNoC: A Learning-enabled, High-performance, Energy-efficient, and Secure On-chip Communication Framework Design

Ke Wang, *Student Member, IEEE,* Hao Zheng, *Student Member, IEEE,* Yuan Li, *Student Member, IEEE,* Ahmed Louri, *Fellow, IEEE*

Abstract—We propose SecureNoC, a learning-based framework to enhance NoC security against Hardware Trojan (HT) attacks while holistically improving performance and power. The proposed framework enhances NoC security with several architectural innovations, namely a per-router HT detector, multi-function bypass channels (MBCs), and a lightweight data encryption design. Specifically, the threat detector uses an artificial neural network for runtime HT detection with high accuracy. The MBCs consist of a router bypass route and reconfigurable channel buffers which can efficiently isolate malicious nodes and reduce power consumption. The proposed data encryption design adapts to diverse traffic patterns and dynamically deploys novel lightweight encryption techniques for desired security goals with improved latency. Additionally, to balance the trade-offs and handle the dynamic interactions of the proposed dynamic designs, a proactive deep-Q-learning (DQL) control policy is proposed to simultaneously provide optimized NoC security, performance, and power consumption. Simulation studies using PARSEC benchmarks show that the proposed SecureNoC achieves 36% higher HT detection accuracy over state-of-the-art NoC security techniques while reducing network latency by 39% and energy consumption by 46%.

Index Terms—Computer Architecture, Network-on-Chip (NoC), NoC Security, Neural Network, Deep Reinforcement Learning

1 Introduction

As technology scales, Network-on-Chip (NoC) architectures [1], [2], have emerged as the prevailing communication fabric for manycore architectures. However, as computing resources are dynamically shared, NoCs are becoming increasingly vulnerable to security threats [3]–[15]. In NoCs, maliciously implanted Hardware Trojans (HTs) [16]–[25] have been shown to destruct NoC functionality, degrade NoC performance, leak information, and covertly transmit data.

A large body of work has been devoted to secure NoCs against HTs from three major aspects, namely HT detection, HT mitigation, and data encryption [16]–[24]. Prior threat detection techniques [3], [5], [19], [21], [24] monitor NoC attributes and detect malicious components by capturing abnormal attribute values (e.g., injection rate, buffer/link utilization, and latency) that far exceed a manually designed threshold. The threshold values, if not carefully selected, can result in inaccurate detection and performance penalties. For HT mitigation, existing designs [11], [12], [24] separate the shared NoC resources (e.g. virtual channels and communication paths) to isolate the HT-infected routers. These techniques can provide non-interference transmissions but inevitably restrict the NoC utilization. Conventional data encryption methods [15], [26]–[28] consist of complex computations and incur additional traffic for broadcasting public keys and sharing private keys, which can result in further network latency and power overheads.

- *Ke Wang, Hao Zheng, Yuan Li, and Ahmed Louri are with the Department of Electrical and Computer Engineering, George Washington University. E-mail: {cory, haozheng, liyuan5859, louri}@gwu.edu*

Manuscript received XX XX, 2021, revised XX XX, 2021.

In this paper, we propose SecureNoC, a learning-based design framework consisting of architectural and algorithmic designs to enhance HT detection, threat mitigation, and data encryption in a holistic manner. We also intend to use machine learning algorithms to optimize the dynamic behavior of the proposed design and provide improved performance and power, as compared to existing techniques. The major contributions of this paper are as follows:

- **Improved Threat Detection Design:** We propose architectural enhancements to the conventional router to improve NoC security with accurate threat detection. The proposed threat detection hardware uses an artificial neural network to accurately identify the HT-injected faults in the transmitted packets and detect HT-infected routers at runtime.
- **Improved Channel Design:** We propose to mitigate HT attacks with a multi-function router bypass channel (MBC) design to route packets and avoid HT-infected routers. The proposed MBC uses a simple switch logic design for reduced network latency and power consumption. Additionally, reconfigurable channel buffers, which can be configured as buffers or repeaters, are implemented at inter-router channels to improve network throughput.
- **On-demand Lightweight Data Encryption Design:** We propose a novel data encryption methodology for on-chip communication named semi-private key encryption (SPK) for data protection. The SPK uses a secret sharing technique for data encryption to reduce the overhead of encryption and improve performance.

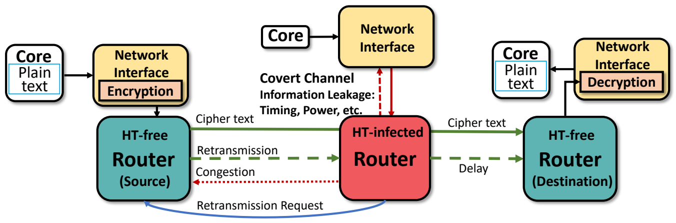

Fig. 1. HT attack model. The packet is transmitted from the HT-free source router to the HT-free destination router via an HT-implanted intermediate router.

- **Learning-based Control Policy Design:** We propose to use deep Q-learning (DQL) to balance the trade-offs and handle the dynamic interactions of the proposed dynamic hardware designs, with the goal of holistically optimizing security, performance, and power consumption. Specifically, the per-router DQL agents learn from the dynamic interactions between the proposed reconfigurable hardware components and the entire NoC environment to evolve an optimal control policy that selects the most suitable operation mode at runtime.

We evaluate the proposed SecureNoC design using GEM5 [29] simulator with PARSEC benchmarks on an 8×8 2D mesh network. Simulation results show that SecureNoC improves HT detection accuracy by 36% and provides advanced data protection over state-of-the-art NoC security techniques [3], [12] while reducing end-to-end network latency and energy consumption by 39% and 46%, respectively.

2 Background and Motivation

Fig. 1 shows packet transmission and the related HT attacks in an 1×3 on-chip network. The figure shows an HT-free source router (green), an HT-free destination router (green), and an HT-infected intermediate router (red). The network interface transforms data generated from the processing cores into packets that consist of multiple flits and inject those flits into the associated routers for transmission. The routers and inter-router links comprise the NoC. Before transmission, the plain-text of sensitive data is first encrypted and turned into cipher-text before being injected into the network. Following that, the network interface generates a packet and injects it into the NoC for transmission. The ciphertext is decrypted at the destination router to recover the original data. In this section, we briefly describe the HT attack model in NoCs.

2.1 Hardware Trojan (HT) Attack Model

Hardware Trojans (HTs) are intentional hardware alterations of the design specification or the corresponding implementation. HTs are implanted during the IC design phase of the circuits. After being implanted, the HTs usually remain dormant to avoid being detected and are activated upon internal or external triggering events. In NoCs, HTs are usually implanted in the routers. Fig. 1 shows the attack model, in which a router is infected with a fault-injection HT. Specifically, HTs can downgrade performance by intentionally injecting faults in packets and incurring retransmission traffic to create network congestion which can significantly disrupt traffic. Furthermore, the congestion can build up excessive back pressure to the upstream routers and saturate the communication channels, thus activating denial-of-service attacks by causing the target router to exhaust scarce resources. According to previous research [11], [14], [30], [31], such resource exhaustion can lead to interference of the transmitted packets of different regions and potentially cause leakage of sensitive information through the side or covert channel effects.

In this paper, we focus on the attack model in which the HT-implanted router insert faults into transmitted packets to inject extra traffic into the network to cause network congestion [6], [11], [12], [32]. We assume the HTs are simple and only able to inject errors into the transmitted packets. In other words, the HTs neither are capable of processing a large amount of data nor altering the functionalities of the HT detection module (e.g. learning the parameters of the Detect-ANN), since adding such functionalities is power and area consuming, which makes the HTs easier to be detected by BIST. However, the HTs can randomize the ratio of the injected faults, so that the characteristic of the altered NoC attributes can not be captured easily. In this paper, existing HTs [7]–[9] with different manufacturing variability are implemented in some of the routers. Specifically, these implanted fault-injection HTs remain dormant until triggered. The triggering events are runtime temperature (router chip temperature) [7], local buffer utilization [8], and operation voltage shifting variation [9].

2.2 Data Encryption Schemes

Encryption methods are used to generate cipher-text to replace plain-text for data protection. Encryption methods can be generally categorized into symmetric encryption and asymmetric encryption. Symmetric encryption uses the same secret key for both encryption and decryption. The key for each data transmission should be unique to ensure security. Specifically, the encryption algorithm produces the cipher-text by taking the key and plain-text as inputs. Similarly, the decryption algorithm uses the ciphertext and the same key to recover the plain text. Asymmetric encryption, on the other hand, uses a set of public keys that are known by all entities. Each router is assigned to a unique public key, and all public keys are stored in a public-key-mapping table which can be accessed by all routers. The source encrypts the data using the destination router's public key, and the ciphertext can only be decrypted by the destination router using a unique private key associated with its public key.

However, in NoC, both encryption methods have potential security vulnerabilities during public key sharing and private key transmission. Conventional complex key sharing techniques solve these problems at the cost of significant computational overhead and power consumption [33]. Consequently, prior works [15], [28] have deployed lightweight encryption schemes. These techniques provide data protection functionalities, such as cryptographic hash function, message authentication, and random number generation to build a data encryption system with reduced overhead.

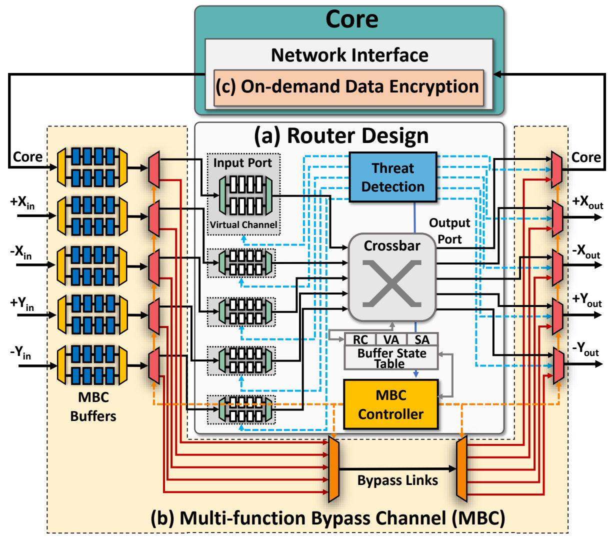

Fig. 2. Microarchitecture of the proposed SecureNoC design. The figure shows (a) the security-enhanced router design, (b) the proposed multi-function bypass channels (MBCs), and (c) the on-demand light-weight data encryption design.

However, these solutions still follow the traditional encryption method that requires multiple rounds of complex operations performed on the plain-text and secret keys, thus incurring overheads in terms of power and timing.

3 Proposed SecureNoC Architecture

We propose SecureNoC for secure and efficient NoC architecture consisting of architectural and algorithmic enhancement techniques, as shown in Fig. 2. On the hardware side, Fig. 2(a) shows the enhanced router where we add a per-router threat detection hardware (blue box inside the router) for accurate runtime HT detection. We also deploy channel buffers [34]–[36] as inter-router buffers for performance enhancement and power savings. We propose new multi-function bypass channels (MBCs) shown in Fig. 2(b) to mitigate HT attacks. These channels are composed of bypass links (red lines) and an MBC controller, which is responsible for traffic flow control and configuration of the channel buffers. On the algorithm side, we propose an on-demand data encryption method with lightweight data encryption algorithms to protect sensitive information with minor overheads, as shown in Fig. 2(c).

3.1 Per-router Threat Detection Hardware

We propose a per-router threat detection hardware which consists of an artificial neural network, named D-ANN [32], for HT detection. Using D-ANN, the threat detection hardware is able to learn from runtime network activities and automatically identify HT-infected routers. Specifically, the threat detection hardware first monitors and extracts the values of local runtime attributes from the on-chip sensors. A total of twelve NoC attributes are used, which include buffer utilization (the number of occupied virtual channels) for each input port (total of 5), link utilization (the value of input-flits per cycle) for each input port (total of 5), and packet injection rate, and local operation temperature. Those attribute values will be fed into the input layer of the D-ANN. Using the input values, D-ANN then calculates a label, either HT-infected or HT-free, as the HT detection result. Afterward, the threat detection hardware forwards the detection result to the proposed DQL module (Sec. 4.2) for operation mode selection.

The proposed D-ANN is a fully connected neural network, which is composed of three layers, namely an input layer, a hidden layer, and an output layer. We explore twelve NoC attributes as inputs, which include buffer utilization (the number of occupied virtual channels) for each input port (total of 5), link utilization (the value of input-flits per cycle) for each input port (total of 5), and packet injection rate, and local operation temperature. The hidden layer, which consists of several ReLU neurons, uses all of these attribute values and calculates the output values of the D-ANN. The number of ReLU neurons in this layer directly impacts the classification accuracy and computational/storage overhead. We implement 30 neurons in this layer for the best accuracy/cost ratio (detailed discussion is given in Sec. 5.4). The output layer consists of two neurons. The first neuron has an output value that equals 0, which means the local router is HT-free, and the second neuron has an output value that equals 1, which means the router is HT-infected. To identify the HT-infected routers promptly while reducing the computational overhead, the proposed D-ANN gathers network attributes and calculates the detection results every 5000 clock cycles.

The objective of the proposed threat detection hardware design is to identify the anomaly behaviors (induced by the HTs) by observing runtime NoC activities (the values of a number of runtime attributes) using the D-ANN. Therefore, we create two data sets for training the D-ANN. One of the two data sets is the training set that consists of the ”normal” runtime attribute values with no implanted HTs. The attribute values are captured at each time step. The other set consists of the ”abnormal” runtime attribute values with implanted HTs. Specifically, first, synthetic and real applications from the PARSEC benchmark (blk, dedup, fre, and swa) are executed in a fully HT-free system for five training rounds while the runtime attributes used in the D-ANN are monitored and recorded. The D-ANN takes these attributes as inputs and calculates the weights as the outputs are all labeled as ”HT-free”, meaning the output neuron is the first neuron (the output value is 0) for all inputs for this data set. Then, the same applications are executed for 20 rounds with HT-implanted routers. The HTs inject transient faults into packets and cause network congestion. During this phase of application execution, runtime attributes are recorded and fed to the input layer of the D-ANN, and the HT-infected routers are labeled in the output layer. Specifically, for the routers where we injected HTs, we set the output neuron of the data set of this phase to the second neuron, and the output value is 1.

During the training process using the two data sets, the D-ANN calculates the outputs using the input attribute values. The calculated outputs will be compared to the assigned output values, so that the error E of the D-ANN model is obtained. The back-propagation algorithm is deployed to use the error E to update the weights W_{ij} for

D-ANN, using the following equation:

$$W_{ij}^{new} = W_{ij}^{old} - \alpha \cdot \frac{\partial E}{\partial W_{ij}^{old}} \quad (1)$$

E is calculated by subtracting the actual output calculated by the neural network and the assigned value (0 for HT-free, and 1 for HT-infected). We update the weights in D-ANNs until they converge. To explore the training sets for a better training result, the HT-infected routers are randomly selected at the beginning of each round. Each round consists of 20 full executions of each synthetic traffic and real application, respectively.

One possible design constraint of the proposed D-ANN is the robustness of the offline-trained model. To tackle this issue, we train the proposed D-ANN with both synthetic traffic and a number of real benchmarks. Moreover, we also train the D-ANN multiple rounds with randomly implanted HTs with the goal of comprehensively covering sufficient communication behaviors. Additionally, the inaccurate detection of HT-infected routers can lead to penalties such as unnecessary router isolation, performance degradation, and insecure data transmission. Thus, a trained model should avoid false-positive results (that identify HT-free routers as HT-infected) and false-negative results (that identify HT-infected routers as HT-free). In this paper, both false-positives and false-negatives can be mitigated by updating and correcting the threat detection results periodically. Specifically, false-positives can be a problem when an HT-free router is always labeled as HT-infected. In the proposed design, even if the HT-free router is labeled as HT-infected by mistake, the detection result will be updated at the next time step. As the trained D-ANN has a high HT-detection accuracy, the wrongly labeled router has a high chance to be labeled as HT-free correctly at the next time step. By doing so, the penalty of isolating that HT-free router will be limited to one time step. Therefore, the false-positive problem can be mitigated. False-negatives occur when the HT is not activated and the router is labeled as HT-free, which are common in conventional designs. The proposed D-ANN resolves this problem by monitoring the runtime NoC behaviors consecutively and providing HT-detection results every 2000 cycles. As the D-ANN utilizes the average attribute values within the time step, it can sensitively capture the anomaly behavior of HT-infected routers, even if the HTs are only activated for a short period. Therefore, false-negatives are reduced.

3.2 Multi-Function Bypass Channels (MBCs)

We propose multi-function bypass channels (MBCs) to bypass traffic around HT-infected routers. We also propose to use tri-state buffers [34]–[37] in the inter-router channels and remove a portion of router buffers to improve network performance and reduce overall power consumption. The design details are described next.

3.2.1 MBC Functionalities

According to the labeling results from D-ANN, the transmitted packets, whose source and destination are both HT-free, are considered as high-security-demand packets, while the packets whose source or destination router is HT-infected are considered as low-security packets. For high-security-demand packets, if no data encryption method is used, it must be ensured that all the routers in the data path should be HT-free. In this case, router bypassing can be used to isolate the HT-infected router during transmission and maintain network connectivity. For this, we propose a multi-function bypass channel (MBC), as demonstrated in Fig. 2. First, MBC integrates a set of bypass links to isolate the HT-infected routers. Additionally, we implement a number of reconfigurable inter-router MBC channel buffers for the use of MBC to improve network throughput. Additional benefits of the MBCs include providing on-demand flit buffering, and enabling router power-gating to save power. Each MBC channel buffer is constructed with tri-state transistors [34] which can be configured as channel buffers or repeaters for different use cases. For example, when the router is HT-infected and to be power-gated for reduced power, the bypass links can be used for transmitting packets, and the MBC buffer can be configured as on-demand link storage buffers for improved throughput. To perform non-interference transmission between high-security packets and low-security packets for improved security, the high-security packets can use MBC to bypass the HT-infected routers, while the low-security packets are transmitted through the router logic of the HT-infected routers. Compared to existing solutions that stall low-security packet transmission while transmitting high-security packets [11] or limit the throughput of different traffic regions [12], the proposed design fully utilize network resources thus minimizing performance losses. At last, we implement an MBC controller for configuring the MBC buffers and controlling the activation of the bypass links.

Specifically, MBC connects all the input and output ports of an isolated router with the bypass links and a simple switch logic using MUXes and DEMUXes. The proposed bypass links allow the high-security packets to be propagated using a round-robin scheme without traversing the HT-infected router logic. Low-security traffic, on the other hand, can still use the router to better utilize NoC resources. Note that the bypass links add five 128-bit data paths to the conventional design. This will ensure the same bisection bandwidth when the router is bypassed and also increase the bisection bandwidth when the bypass links are used along with the router links. The overheads of these additional links are demonstrated in Sec. 5.5. The mode selection of the channel buffers is controlled by the MBC controller with a 1-bit mode-selection signal. The benefit of using the reconfigurable channel buffers is as follows. First, MBC channel buffers can buffer the high-security demand flits that bypass the HT-infected router to improve MBC throughput. Second, for high traffic loads, the channel buffers can be used to buffer the incoming flits in addition to router buffers. Compared with router buffers, the channel buffers significantly reduce power consumption with negligible latency overhead [37]. Third, for moderate traffic loads, the channel buffers can be configured as repeaters to reduce network latency. Finally, when the traffic load is low, the under-utilized router can be power gated. In this case, with the additional channel buffers, MBC allows multiple sporadic flits to be propagated without first powering on the corresponding router, which can achieve significant power savings. The dynamic selection of MBC functionalities is explained in Sec. 4.1. Next, we detail the buffer allocation for both channel buffers and router buffers, as well as the flow control algorithm.

3.2.2 Channel Buffer Allocation and Flow Control

In this paper, we use the unified dynamic buffer allocation scheme [38] for both MBC channel buffers and router buffers to maximize network throughput. The proposed per-router buffer allocation table records the routing information of all five input ports of the corresponding router and can be accessed consistently, even when the router is bypassed, isolated, or power-gated. The proposed unified dynamic buffer allocation table is constructed by adding several new entries to the existing virtual channel (VC) state table [39]. The entries of the proposed buffer allocation table include the VC identifier (VC), read pointer (RP), write pointer (WP), allocated output ports (OPX and OPY), output VC (OVC), state (ST), credit count (CR), and four MBC-related entries, namely an input port identifier (Port) that indicates the input port ID of the incoming flit, a downstream router status indicator (DRS) that records the availability of the downstream router, a channel buffer pointer (CBP), and a channel buffer credit indicator (CBC).

The flow control using the proposed buffer allocation table is as follows. First, the routing information in the header flit is extracted and used for route computation and virtual channel (VC) allocation. When the router is HT-free and available for packet transmission, the buffer allocation table assigns an unoccupied VC to the header flit. When a free VC slot is assigned to the header flit of a packet, the credit count (CR), which records available router slots (including router/VC buffers and channel buffers), is updated. The allocated VC slot, along with the computed OPX/OPY and OVC, are recorded in the buffer allocation table. Afterward, the body flits use the recorded VC of the header flit and the corresponding output information to complete the packet transmission. It is noteworthy that if the router is bypassed, isolated, or power-gated, incoming packets can still be transmitted with no problem following the same process, as the buffer allocation table is accessible. Since the simple switch design using MUX/DEMUX has limited throughput, a routing mechanism that avoids transmitting intense traffic through bypass channels should be applied. Moreover, to better utilize network resources, especially the isolated routers, a different routing algorithm might be needed for the low-security packets. Thus, we propose an adaptive routing algorithm [32] that intelligently balances traffic loads with various routing algorithms (O1TURN, West-First, and Negative-First) to avoid injecting into bypass channels and optimize the worst-case throughput of different NoC traffic patterns using various routing algorithms [40]–[42]. The O1TURN routing dynamically applies XY or YX routing for each packet to better utilize the network spatially under normal traffic loads. The West-First and Negative-First restrict different types of turns that are allowed and achieve lower latency and less dynamic power consumption than O1TURN under intense traffic-loads [40]. The routers in the proposed design have multiple virtual channels to avoid both protocol and routing deadlocks.

3.3 On-demand Light-weight Data Encryption using a Semi-private Key Sharing (SPK) Method

In data-protected NoC systems, flits are encrypted before being injected into the local router. Typical data encryption methods consume excessive computation time and power consumption due to the multi-round complex computations. For example, conventional AES encryption uses a 128-bit secret key for a 128-bit flit and consists of 10 rounds of computation, which can cause significant performance degradation in resource-constraint NoC systems. Previous works [15], [43] have reduced the overhead of the traditional encryption techniques with reduced encryption rounds, proposed smaller block/key sizes, or optimized hardware designs. However, these techniques still have complex computations that consume multiple clock cycles. Moreover, the static use of these techniques can lead to additional costs or security issues as described previously. For example, deploying complex data encryption at every hop can result in excessive power consumption as the packets injected by the HT-infected routers need not to be protected. Similarly, simple encryption that has reduced overheads might not be sufficient for systems with multiple HT-infected routers.

To this end, we propose an on-demand light-weight data encryption technique that adapts to diverse traffic for optimized data protection while reducing power and computational overheads simultaneously. Specifically, for low-security traffic whose source or destination is detected to be HT-infected by D-ANN, no encryption algorithm is applied. For high-security-demand traffic, we propose a lightweight encryption method called semi-private key (SPK) encryption for data protection. The proposed SPK is based on the secret-sharing theory. The proposed SPK eliminates the multi-round complex computation of the conventional AES and only requires a small number of additions and multiplications. Moreover, by using a semi-private key that is stored within each router, the proposed SPK eliminates both the traffic for private key transmission and the security vulnerability of using fully public keys as described in Sec. 2.2.

In the proposed SPK, a single public key is used for both encryption and decryption of all transmitted packets. We use the secret sharing [44], [45] theory to only store a unique *share* of the public key in each router, and the public key can be recovered with multiple unique shares. The secret sharing theory [44], [45] is based on the *Lagrange interpolating polynomial* with a (t, n) threshold. A certain secret S is shared by n participants in the system, each of which keeps a unique *secret shadow* of S. The secret S can only be recovered if at least t $(t \leq n)$ secret shadows are retrieved. The secret sharing theory is proven to be absolutely secure in the mathematical finite field [46]. Specifically, to protect the secret S, first, a Lagrange polynomial is constructed as follows:

$$L(x) = S + a_1 \cdot x + a_2 \cdot x^2 + ... + a_{t-1} \cdot x^{t-1} \mod (p) \quad (2)$$

where S is the secret. p is a pre-defined prime number which is larger than S. Note that the degree of $L(x)$ is $(t-1)$, meaning that any attempt to reconstruct the polynomial with less than t secret shadows will give the incorrect polynomial with the wrong secret S.

The secret shadow s_i, where $i \in (1, t-1)$, is then created using different integer values of x_i and calculating the corresponding $f_i(x)$ values. Next, the value P and a unique shadow $s{+}i = (x_i, f_i(x))$ is sent to each router. After distributing the secret shadows, the Lagrange polynomial $L(x)$ is destroyed by deleting the secret S and coefficients a_i, where $i \in (1, t-1)$. To retrieve the secret S, at least t secret shadows should be compromised. With t secret shadows, S can be recovered using:

$$S = F(0) = \sum_{i=1}^{t} \frac{f_i(x)\prod_{1\leq j\leq t, j\neq i}(x-x_j)}{\prod_{1\leq j\leq t, j\neq i}(x_j-x_i)} mod(p) \quad (3)$$

In this paper, we use the same concept while extending the (t, n) threshold to a set of (k, n) thresholds for different routers to share the public key K in the NoC system. The variables of Lagrange polynomial $L(x)$ are randomly selected, while the value of n equals the number of routers. The computation of $L(x)$ and the corresponding secret shadows are completed in a least-utilized HT-free core. Later, the secret shadows, or semi-private keys, are distributed to each router. As each router only stores a share of the public key, we call this method "semi-private key sharing". To retrieve the original secret key S, the router needs to compromise the other routers and gather at least k other secret shadows.

The selections of the polynomial and the variable t can impact the network performance. For a larger t, the routers in the NoC system need to gather more secret shadows, which can result in increased traffic loads and additional computation time. A smaller t, on the other hand, requires fewer secret shadows to recover the secret key, thus may pose security risks. To achieve the optimized data protection with minimized overhead, we define the k value for each router as $k = d_i + 1$, in which d_i is the degree of the router R_i. The $t-k$ secret shadows are stored locally, while k secret shadows are distributed among other routers. For example, in a 2-D mesh network, the k value for the routers in the corners is 3. Similarly, the k values for the routers on the edges and in the center of the mesh are 4 and 5, respectively. Additionally, We only authorize the source and destination router to gather the secret shadows from adjacent routers. By doing so, the source router and the destination router can gather sufficient secret shadows, and the HT-infected routers are prevented from retrieving the secret key. Note that the proposed D-ANN and MBC ensure that the HT-infected router will not become the source or destination for traffic with high security demands. Besides, as discussed previously, there is no encryption or decryption needed for low-security traffic, thus the HT-infected router is neither authorized to acquire secret shadows from other routers nor to get secret shadows from the key distribution phase.

According to information-theoretical entropy function [47], the entropy of the proposed SPK is $H(S|S_T) = H(S)$, which is perfect privacy for each transmission [48]. Therefore, SPK can fulfill data protection demands with reduced overheads.

Fig. 3 shows an example of the secret shadow gathering in the proposed SPK. The packet is transmitted from the source router to the destination router, and the data path is shown with blue arrows, using YX routing with an HT-infected router located in the data path. Similar to the itera-

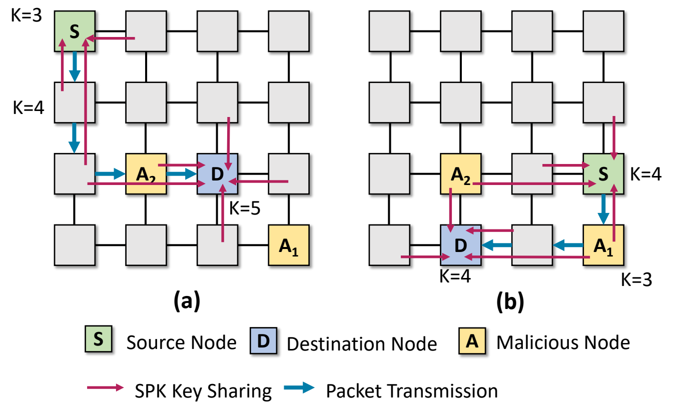

Fig. 3. Examples of data encryption using SPK.The blue arrows show the data path of the packet transmitted from the source router to the destination router. The red arrows show the data transmission for sharing secret key shadows.

tive operations of D-ANN, the secret shadows are created and assigned in discrete time steps. At the beginning of each time step, the Lagrange polynomial with a new public key S is decided, and the secret shadows are transmitted to each router, as shown with red arrows. During packet transmission, the source and destination router first issue a handshake signal to the adjacent router and selects one additional router in its X or Y direction using the XY/YX routing algorithm. Subsequently, the source router gathers the secret shadows and retrieves the secret key for encryption. The encryption method used is a single-round XOR operation with S for minimized overhead. Similarly, the destination router can calculate the key and use it for decryption. As described in Sec. 2.1, an HT-infected router might access the secret shadows of the adjacent router through information leakage at this stage. However, it is insufficient for secret key recovery.

The use of SPK ensures data protection and eliminates the multi-round complex computation of conventional encryption methods and full-key transmission in state-of-the-art solutions [15], [28]. The overheads of the SPK are induced by key computation and transmission of the secret shadows. Detailed performance analysis, in terms of security, power consumption, and computation overheads, is presented in Sec. 5.

4 Dynamic Operation Modes and Learning-Based Control Policy

4.1 Dynamic Operation Modes

We propose three dynamic operation modes, each of which has various security techniques and hardware configurations. The operation modes will be selected and deployed by each router independently and iteratively in a sequence of discrete time steps, directed by a deep-Q-learning (DQL)-based control policy. The proposed operation modes are described as follows.

- **Operation Mode 1:** In this operation mode, the router is power-gated, and the associated MBC bypass links are activated. When configured as such, the MBC channel buffers are configured to be on-demand link storage for the incoming flits. This mode is triggered when the router is underutilized or when the corresponding router is detected as an HT-infected router. In this case, the router is powered off and isolated to significantly reduce overall power consumption and provide network security.

- **Operation Mode 2:** In this mode, the bypass channel is activated, while the router is not power-gated. In this case, the MBC bypass links are used to provide non-interference packet transmission. Specifically, the bypass channel dynamically routes high-security packets without traversing HT-infected routers, as well as utilizing the bypassed routers to propagate low-security packets without degrading network performance. MBC channel buffers are configured as flit buffers. This operation mode provides basic network security as data protection techniques are not deployed.
- **Operation Mode 3:** In this mode, the MBC bypass links are disabled. The router switches its adaptive encryption hardware to SPK. This operation mode is beneficial when the transmitted data requires moderate protection aside from non-interference packet transmission. The MBC buffers are configured as repeaters to reduce latency.

The three proposed operation modes work together to comprehensively cover broad use cases with different traffic, workload, and security requirements. The three operation modes each has various security-enhancement techniques and hardware configurations. Specifically, the first operation mode, which power-gates the HT-infected router, can provide basic security features. This is beneficial when the traffic load is low, as the power-gating feature can save static and dynamic power consumption. However, if the traffic load is high, as this operation mode only activates the bypass links for data transmission, this operation mode will lead to excessive network latency. Therefore, we design the second operation mode, which allows the low-security packets to be transmitted using the HT-infected routers. This is beneficial to the network latency when traffic load is high, while still providing basic security due to the non-interference transmission. For the maximized security, we propose the third operation mode with an additional data encryption feature. This operation mode sacrifices performance and power, as it induces additional timing and power overheads for data encryption, for the highest level of data protection. In summary, Operation Mode 1 is beneficial when traffic load is low and basic security feature is required; Operation Mode 2 is beneficial when traffic load is high and basic security feature is required; Operation Mode 3 is beneficial when maximum security feature is required.

The dynamic operation modes are independently yet simultaneously selected by each router using a deep-Q-learning (DQL)-based control policy demonstrated in Sec. 4.2 at each time step, and the buffer allocation table is updated accordingly. It is noteworthy that even though the MBC configuration cannot be changed within a time step, the encryption method (bypass-only or SPK) suggested by the per-router DQL can be overwritten by the HT-free cores on demand. However, this overwriting function is disabled in this paper during design evaluations in Sec. 5. As previously discussed, the proposed on-demand data encryption and per-router power-gating/bypassing design allows individual routers to deploy the most beneficial strategy at any time step, thus achieving improvements on performance metrics for the entire NoC.

4.2 Deep-Q-Learning (DQL)-Based Control Policy

We present a per-router deep-Q-learning (DQL)-based control policy to balance the trade-offs and select one of the dynamic operation modes at runtime. DQL [49] is a type of reinforcement learning [50] algorithm that learns from the dynamic interactions between autonomous agents (routers) and the environment (NoC system) and optimizes the router control policy. In SecureNoC, each per-router DQL agent interacts with the NoC system in a sequence of discrete time steps t.

At each time step, per-router DQL agent monitors a set of runtime network attributes and observes the current working status, named *state* s. With the given state s, the router takes an *action* a by selecting an operation mode that will be applied at the following time step. The selection of the action a is directed by a trained policy, which indicates the long-term return $\mathcal{R}_t$ of taking the action a. At the subsequent time step, the router impacts the entire NoC system after taking the selected action by incurring changes of NoC attributes and performance metrics. Comparing the new performance metrics to their previous values, the network will have a system-level evaluation on the immediate benefit r of taking the action a. The value r is used to update the long-term return $\mathcal{R}_t$ for decision making in the future. Meanwhile, a new state s' is observed, and a new time step will start.

In DQL, the goal of the DQL agent is to learn a policy that optimizes the agent's long-term *return*, $\mathcal{R}_t$, which is the exponentially discounted sum of the immediate rewards of all future time steps [?], [36], [51]–[53]. The return at time step t is therefore defined as:

$$\mathcal{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4}$$

The variable γ (where $0 \leq \gamma \leq 1$) in this equation 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 increasing weight to future rewards. The impact of γ on DQL is presented in Sec. 5.4.

In this paper, we use the deep-Q-learning algorithm [35], [50], [54] to estimate $\mathcal{R}_t$ with an *action-value function* $Q(s, a)$. The Q-value represents the expected maximum long-term return that the agent, starting in state s, follows the optimal policy for all future actions. The design details of the proposed DQL are presented as follows.

Action-Value Function and Deep-Q-learning In DQL, we model the return $\mathcal{R}$ as follows. In this paper, a *model* of the environment 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. The model is specified through a probability distribution $p(s_{t+1}, r_{t+1}|s_t, a_t)$. Correspondingly, the agents compute an action-value function Q^{π}(s, a) that captures the return $\mathcal{R}$ 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:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_p\{\mathcal{R}_t | s_t = s, a_t = a, a_{t+1:\infty} \sim \pi\} \\ &= \mathbb{E}_p\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... \\ &\qquad | s_t = s, a_t = a, a_{t+1:\infty} \sim \pi\} \\ &= \mathbb{E}_p\{r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \\ &\qquad | s_t = s, a_t = a, a_{t+1} \sim \pi\} \end{aligned} \tag{5}$$

Because the agent always intends to maximize its expected return, it can be shown that in the optimal policy π^*, the optimal Q-value of a given pair of state-action is:

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}_p\{R_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\} \\ \pi^*(s) &= \max_a Q^*(s,a) \end{aligned} \tag{6}$$

The first equation in (6) is called the Bellman optimality equation and is used to formulate RL algorithms such as the Q-learning algorithm used in this paper.

To find the optimal Q-value function *Q*(s, a)*, tabular Q-learning algorithm [55] can be used. In conventional Q-learning, assuming the state s is discrete, a table of Q-values is initialized with random values 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. After taking an action *a* and observing the reward *r* and new state s', the action-value table entry *Q(s, a)* is changed using the following temporal difference rule:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s', a')] \tag{7}$$

The learning rate α can be reduced over time and determines how well Q-learning will converge. With an appropriate value of α, Q-learning converges to the optimal Q-value function Q* and its corresponding optimal policy π [55]. In the proposed DQL, we replace the Q-table with a trained neural network, called Q-ANN. The design details of DQL are presented below.

Design Space of the DQL: We formulate the design space of the proposed DQL as follows:
<u>State Space:</u> We construct the state vector S with a set of local NoC attributes for each router, which are observed at each time step, as listed in Table. 1. These attributes include input-link-related metrics, buffer-related metrics, output-link-related metrics, and security metrics. According to the range of each attribute, all of these attribute values are discretized into several bins to limit the number of (s,a) pairs. For example, link utilization is discretized into 5 levels. We select these listed attributes because they represent the traffic patterns and congestion-related NoC behaviors. Moreover, global information, such as dimensional traffic intensity, is not selected to reduce the overhead of collecting the attribute values at runtime.
<u>Action Space:</u> The action space $A = \{a_1, a_2, a_3\}$ consists of the three operation modes, as described in Sec. 4.1. Specifically, a_1 represents the Operation Mode 1, and a_2 represents the Operation Mode 2. a_3 represents the Operation Mode 3.

TABLE 1
Attributes in the *state* vector.

Category	Attributes	Description
Router Input Related Metrics	1. +X link utilization	Flits/cycle of +X input port
	2. –X link utilization	Flits/cycle of –X input port
	3. +Y link utilization	Flits/cycle of +Y input port
	4. –Y link utilization	Flits/cycle of –Y input port
	5. Local port injection rate	Injection rate in flit/cycle
Buffer Related Metrics	6. +X buffer utilization	Number of utilized buffers at +X input port
	7. –X buffer utilization	Number of utilized buffers at –X input port
	8. +Y buffer utilization	Number of utilized buffers at +Y input port
	9. –Y buffer utilization	Number of utilized buffers at –Y input port
	10. Local buffer utilization	Number of utilized buffers at local input port
Router Output Related Metrics	11. +X Link utilization	Output flits/cycle of +X output port
	12. –X Link utilization	Output flits/cycle of –X output port
	13. +Y Link utilization	Output flits/cycle of +Y output port
	14. –Y Link utilization	Output flits/cycle of –Y output port
	15. Local port link utilization	Output flits/cycle of local output port
Security Related Metrics	16. +X D-ANN result	D-ANN label of +X downstream router
	17. –X D-ANN result	D-ANN label of –X downstream router
	18. +Y D-ANN result	D-ANN label of +Y downstream router
	19. –Y D-ANN result	D-ANN label of –Y downstream router
	20. Local D-ANN result	D-ANN label of local router

<u>Reward Function:</u> At time step t for router i, the immediate reward function is designed as:

$$r_{i,t} = -\log_a(Latency_{i,t}) - \log_b(Power_{i,t}) \tag{8}$$

The $Latency$ is the network latency of router i, which is the average end-to-end packet latency calculated by the average timing difference between the packet injections and ejections within the time step. The $Power$ refers to the average power consumption that equals the summation of the static and dynamic power consumption monitored by the NoC's shared power module. The coefficients a and b adjust which one is the prior optimization objective over the other. In this paper, both a and b are set to 2.

The Working of DQL: In DQL, a *Q(s, a)* value is used to estimate the long-term return of taking a specific action. The Q-value is defined as:

$$Q(s,a) = (1-\alpha) Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s', a')\right] \tag{9}$$

The value $\max Q(s', a')$ represents the maximum Q-value in the state entry s', which is obtained at the beginning of the following time step. The coefficient α is the learning rate of DQL. It determines how much the intermediate reward impacts the expectation of the long-term return. It is also noteworthy that the selection of the learning rate α also impacts the convergence time of DQL and the performance of the trained policy. A detailed discussion of how the coefficients impact system performance is presented in Sec. 5.4.

In the proposed design, the Q-values are calculated by an artificial neural network, called Q-ANN, at runtime. In this paper, each Q-ANN consists of an input layer with 20 neurons, a hidden layer with 30 neurons, and an output layer with 3 neurons. The input layer represents the selected NoC attributes, while the output layer includes the Q-values of the three possible actions to take. The hidden layer and weights in the Q-ANN record the correlation between the observed state and the corresponding Q-values. This eliminates the excessive area overhead of conventional designs [50], [56] that store the Q-values in state-action mapping tables.

Fig.4 shows the working of the proposed DQL-based control policy during the execution of an application. First, all Q-values are initialized to 0, and the operation modes for

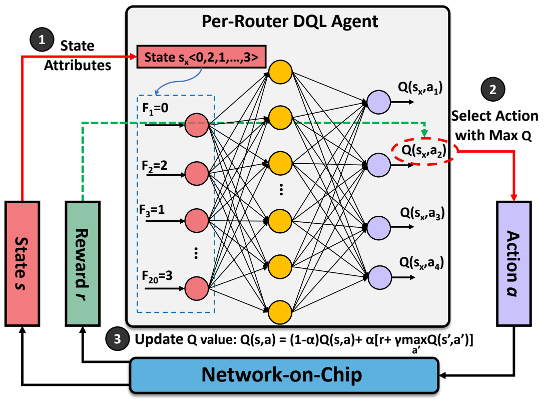

Fig. 4. Deep-Q-learning process. At time step t, the router observes the state, and the action with the maximum Q-value for the current state is selected. Thereafter, the reward for $r(s, a)$ is calculated, and the Q-value is updated following (9).

all routers are set to Operation Mode 2. At each time step, the DQL process consists of a number of stages, as shown in Fig. 4. At stage ❶, the DQL agent (router) monitors the values of runtime NoC attributes $F_1, F_2, ..., F_{20}$. These attribute values together formulate a state vector s_x, which is the current state vector for the DQL agent. In conventional Q-learning, a Q-table is used to map the state vector s_x to the Q-values of all possible actions. In the proposed DQL, we replace the Q-table with a trained neural network, Q-ANN. In the proposed DQL design, these values, or the state vector, are fed into the Q-ANN's input layer. The Q-values of all possible actions for s_x are calculated by the Q-ANN. Afterwards, at stage ❷, the router selects an action a that has the highest Q-value from the three possible actions. The selected action will be deployed at the next time step (for example, a_2). At the following time step, at stage ❸, as the router takes the selected action a_2, the router interacts with the entire NoC system, which result in a transition to a new state s'. Meanwhile, the NoC performance metrics (i.e., network latency and power consumption) are captured, and a reward r is calculated. The reward will be used in the temporal difference rule shown in (9) to update $Q(s, a)$. Each router will go through the three stages at each time step. Note that stage ❸ can be skipped when the Q-ANN is stable and converged. In the proposed design, we use an offline-training process, therefore the Q-value updating stage is only used for training and does not exist in the inference phase. The offline-training process is described below.

Training and Testing Phases of the Proposed DQL: In this paper, we first train the proposed DQL offline with GEM5 [29] simulator with a set of synthetic traffics and real applications from the PARSEC benchmark (blk, dedup, fre, and swa). At first, Q-values for all possible (s, a) pairs are initialized to zero. During the training phase, a subset of PARSEC benchmark applications and synthetic traffic are executed using GEM5, and the dynamic interactions between the routers and the entire NoC system are recorded. The router observes the current s_x at each time step and selects an action that has the highest Q-value for the specific s_x. If several actions are with the same Q-values, a random action will be selected. As we designed the reward using a negative log-function, all of the visited state-action pairs have negative Q-values, while the Q-values for all the unvisited (s, a) pairs are zero. Therefore, the DQL agents are forced to explore new (s, a) pairs at the early stage of the training process. Moreover, an exploration factor ϵ [50], [56] is used to allow each router to take a random action with a small probability of ϵ at each time step. This further increases the exploration of DQL to avoid sub-optimal decisions. Upon taking the action, the router observes a new state and the variation of system performance metrics. During this process, the Q-ANN uses the attribute values of the state vector as inputs of the input layer, and the outputs of the Q-ANN are the Q-values. The Q-ANN uses (9) to calculate the difference ΔQ at each time step. ΔQ is then be fed to the mini-batch gradient descent as the error of D-ANN output. A back-propagation is deployed to update the weights in the hidden layer.

The offline-trained model is capable of being applied to different applications without a new training procedure. Specifically, the proposed training process uses a number of synthetic traffic and carefully selected real applications from the PARSEC benchmark (blk, dedup, fre, and swa) to abstract a wide range of data and traffic patterns to comprehensively cover the communication behaviors. These traffic patterns are captured by the NoC attributes and can cover as many common communication patterns in the testing phase. The proposed DQL makes decisions only rely on the captured communication patterns, or observed attribute values, regardless of what specific application is used. Therefore, the trained model can be directly utilized without retraining, as the dynamic behaviors of the testing application can also be destructed into the communication patterns captured by the trained model.

The trained ANN will be implemented in the routers for testing, without further updating to reduce computational overheads. In this paper, since we only focus on fault injection HTs, the training result is acceptable as long as the ANN can distinguish injected faults and apply suitable security mechanisms based on the NoC attributes. Therefore, there is no need to retrain the ANN for different applications. In the testing phase, the DQL agent observes the attribute values, calculates three Q-values, and selects the action that has the highest Q-value. The timing and area overheads of the proposed DQL are presented in Sec. 5.5.

5 Experimental Results

5.1 Simulation Setup

We evaluate the proposed SecureNoC design using the GEM5 full-system simulator [29], in which we fully incorporate the HT attack models and security techniques. Table 2 describes the simulation parameters used.

Before evaluation, we first train the proposed D-ANN and DQL (Q-ANN) using synthetic traffic and real applications that are different from the testing data set. The Q-values are initialized to 0. The learning rate α and the discount rate γ are set to 0.1 and 0.9, respectively. Additionally, the DQL agents have a small probability of $\epsilon = 0.05$ to select a random action for exploring unvisited state-action pairs. The trained D-ANN and Q-ANN will be implemented in

TABLE 2
Simulation environment setup.

Field	Value
Processing Units	64 CPU Cores @ 32 nm, 1.0V/2.0GHz
NoC Configuration	8 × 8 2D Mesh network, 4-stage router
Packet Parameter	128-bit per flit, 4 flits per packet
Access Delay (cycle)	4 cycles to L1 cache (64KB) 8 cycles to L2 cache (8MB) 160 cycles to main memory
Techniques	Baseline: FHL [3]+ AES [57] + isolation
	FHL [3] + AES [57] + NIBR [12]
	D-ANN + AES [57] + NIBR [12]
	D-ANN + SPK + NIBR [12]
	Proposed SecureNoC Design: D-ANN + SPK + MBC/DQL

each router for the testing phase. Subsequently, we randomly select 6 routers in the NoC to be HT-infected routers. At runtime, these HT-infected routers can inject packets that cause over 70% link and buffer utilization.

During the testing phase, the PARSEC benchmark suite [58] is tested, with the time step length set to be 5000 cycles. Each router is initialized to use operation mode 2. The performance of the proposed SecureNoC design is compared to four other solutions, which are listed in Table 2. Note that the Proposed SecureNoC design using MBC replaces half of the router buffers. or VC buffers, with the proposed reconfigurable channel buffers. Specifically, in all other techniques listed in Table 2, each input port has a total of 16 VC buffers, while SecureNoC has 8 channel buffers and 8 VC buffers for each port. This allows a fair comparison across all techniques. The testing phase for each benchmark application lasts for the full execution time of each application. We evaluate the area and power consumption with Synopsys. The value of power consumption refers to the summary of NoC static power and dynamic power using Synopsys and DSENT, respectively. Specifically, We first modeled the static power of all components with Synopsys. Afterward, the modeled parameters are fed to the power model of GEM5 (DSENT). At runtime, during the execution of different PARSEC applications, DSENT calculates the dynamic power consumption by monitoring NoC activities (i.e., buffer-writes, crossbar, virtual channel and switch, D-ANN calculations, and DQL agent activation).

5.2 Performance Analysis

Speedup: We calculate the speedup by comparing the full-application execution time of each benchmark application when using a selected technique (FHL+AES+NIBR, D-ANN+AES+NIBR, D-ANN+SPK+NIBR, or the proposed SecureNoC) against the execution time when using the baseline technique, as shown as the following equation:

$$Speedup = \frac{ExecutionTime_{baseline}}{ExecutionTime_{technique}} \tag{10}$$

Since we use offline training, the training time is not included in the execution time above. The simulation result is shown in Fig. 5.

As can be seen in Fig. 5, simply deploying the NIBR technique for HT mitigation has limited speedup because

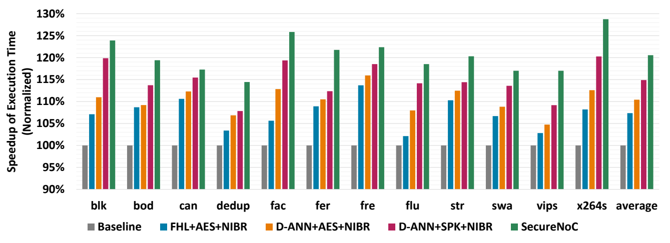

Fig. 5. Speedup of full application execution time comparison, normalized to the baseline (higher is better).

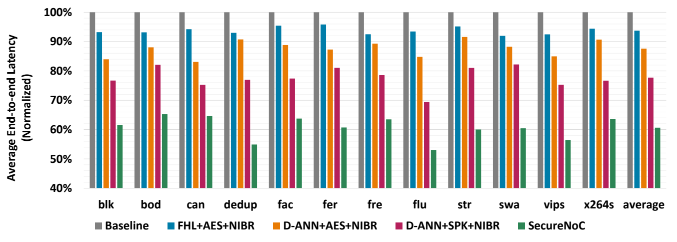

Fig. 6. Average end-to-end latency comparison, normalized to the baseline (lower is better).

of the limited throughput. SecureNoC achieves an average of 21% speedup over the baseline, due to its capability of selectively applying data protection mechanisms. The results are showing similar trends across all applications because the majority of the execution time comes from the computational overheads of the encryption methods. The use of D-ANN improves the execution time since FHL can induce more false-negatives, which can result in more HT-injected faults and retransmission traffic. However, as the routers can be idle, which means the traffic intensity is not always high during the execution, the differences in speedups of using different techniques are not as significant as they are shown in the rest metrics (latency, power, and energy).

Network Latency: We evaluate network latency using the average end-to-end packet latency, as shown in Fig. 6. It can be seen, as compared to the baseline, the use of FHL+AES+NIBR and D-ANN+AES+NIBR achieve 6% and 12% end-to-end latency reduction, respectively, because NIBR and AES induce computing overheads during encryption and routing computation, respectively. However, it can be indicated that even though the routing algorithms NIBR have longer computation time, they still benefit the overall network latency, because network irregularity is avoided. SecureNoC achieves 39% end-to-end latency reduction on average, thanks to the use of the lightweight encryption method and better-utilized network resources during packet transmission. The comparison between D-ANN+SPK+NIBR and SecureNoC justifies the benefit of using DQL for HT mitigation. It shows that the machine learning-based solution (DQL) achieves 27% end-to-end latency reduction, as SecureNoC can adapt to different traffic loads and proactively deploy the most suitable operation modes.

Static Power Consumption: We evaluate the static power consumption as shown in Fig. 7. It is shown that FHL+AES+NIBR and D-ANN+AES+NIBR increase the static power consumption, as compared to the baseline,

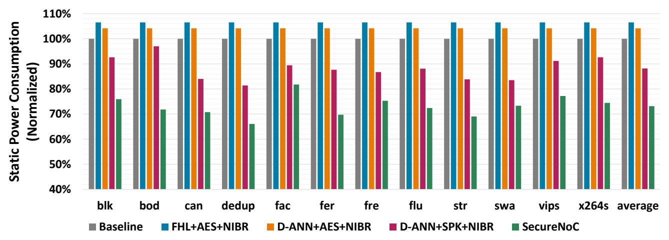

Fig. 7. Static power consumption comparison, normalized to the baseline (lower is better).

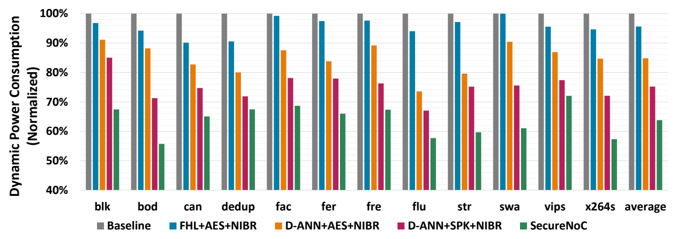

Fig. 8. Dynamic power consumption comparison, normalized to the baseline (lower is better).

because of the complex routing computation and hardware implementation of the neural networks. The use of SPK reduces the static power consumption by 12% due to the simplified cryptography hardware. The proposed SecureNoC using D-ANN, SPK, and MBC achieves 27% static power reduction, due to the use of router power-gating and replacing half of the power-consuming router buffers with MBC channel buffers.

Dynamic Power Consumption: The dynamic power consumption comparison is shown in Fig. 8. It can be seen that FHL+AES+NIBR achieves the least dynamic power reduction due to the complex AES computation and routing computation. The use of SPK eliminates the complex computation of AES and achieves dynamic power reduction. The proposed SecureNoC achieves the highest dynamic power consumption reduction, which equals 36%. The dynamic power reduction of the proposed design is achieved by the use of channel buffers, as well as reducing the computation of data encryption.

Energy Consumption: We define energy consumption as:

$$Energy = (P_{static} + P_{dynamic}) \times T_{exec} \tag{11}$$

P_{static} and $P_{dynamic}$ are static and dynamic power consumption, respectively. T_{exec} is the execution time of each benchmark application. Fig. 9 shows the energy consumption of all applications using different techniques. It shows that SecureNoC reduces energy consumption by 46%, which outperforms other techniques with no machine learning. This is because SecureNoC can improve both static and dynamic power consumption, as shown previously.

5.3 Security Analysis

HT Detection Accuracy: The HT detection accuracy is calculated with the ratio of the number of correctly identified HT-infected routers within one time step. Fig. 10(a) shows that the proposed D-ANN achieves 96% accuracy with 30 and more neurons in the hidden layer. With high detection

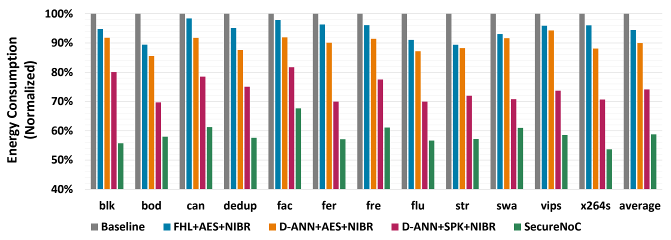

Fig. 9. Energy consumption comparison, normalized to the baseline (lower is better).

accuracy, security threats can be efficiently mitigated using the proposed security techniques.

Data Protection Analysis: The proposed SecureNoC design first enhances security with MBC. Similar to previous designs [11], [12], our design provides non-interference packet transmissions that can prevent sensitive information leakage via the covert channels induced by HTs. We then propose SPK for data protection. The security of transmitted data is preserved by the (t, n) threshold discussed in Sec.3.3. As long as the intermediate router cannot gather t secret shadows, the original message will not be recovered. In our design, the SPK protects transmitted packets that have HT-free source and destination routers. Our design ensures that only the source and destination routers can gather sufficient secret shadows and construct the encryption key. In this case, intermediate routers can only covertly gather $(t-1)$ secret shadows in maximum. Moreover, the proposed HT detection ensures that the HT-infected routers are not authorized to encrypt/decrypt data or gather secret shadows when they are the source or destination routers. Therefore, HT-infected routers have minimal chances to retrieve the public key. Additionally, the keys used in SPK are changed at each time step for security.

Therefore, our approach can achieve secure on-chip communication by accurately detecting and mitigating HTs, as well as providing data encryption with protected secret keys.

5.4 Sensitivity Study

Impact of the Hidden Layer Size of D-ANN: We study the impact of the hidden layer size of D-ANN on the detection accuracy by varying the number of the hidden layer neurons, as shown in Fig. 10(a). Fig. 10(a) shows that implementing more neurons in the hidden layer results in higher D-ANN detection accuracy. It also can be indicated that a single-hidden-layer construction performs better than multi-hidden-layer construction because the former design can mitigate overfitting and is sufficient to deliver the desired accuracy. Therefore, in this paper, we use a single-hidden-layer design with 30 neurons in the hidden layer.

Impact of Time Step Length: Fig. 10(b) shows the impact of time step length on overall performance, where we run the evaluation tests with the time step set to 2000,5000, 10000, 50000, and 100000 clock cycles, respectively. As shown in Fig. 10(b), the proposed DQL control policy using a 5000-cycle time step achieves the best network latency and energy consumption. That is because a shorter time step, even

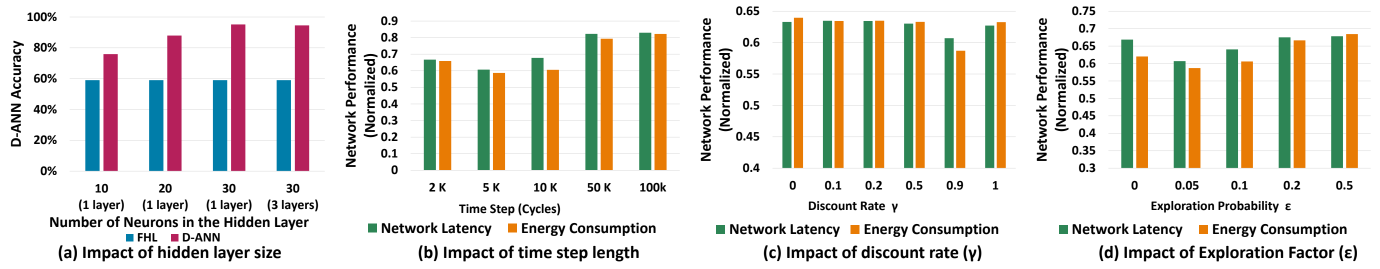

Fig. 10. Impact of (a) hidden layer size, (b) time step length, (c) discount rate γ, and (d) exploration factor ϵ of DQL on network performance metrics.

though it provides finer-grain control, can lead to performance degradation (as the timing and power overheads of DQL computation are more pronounced), while a longer time step only provides coarse-grain control.

Impact of Discount Rate γ : As discussed in Sec. 4.2, the discount rate γ determines the impact of future rewards on the total return. In this test, we varied the value of γ to find out the most suitable γ for SecureNoC, as shown in Fig. 10(c). As shown in Fig. 10(c), the highest performance improvement is achieved with γ is set to 0.9. This is because a smaller γ can lead to a near-sighted control policy, which cannot provide long-term benefits, while a larger γ can lead to convergence failure thus resulting in performance degradation.

Impact of Exploration Factor ϵ : The exploration factor ϵ, which can be set to values from 0 to 1, indicates how often the DQL agent explores unvisited state-action pairs regardless of the Q-values. In this test, we explore how ϵ can impact system performance, which is shown in Fig. 10(d). As can be seen in Fig. 10(d), SecureNoC achieves the lowest network latency and energy consumption when ϵ is 0.05. The reason is that when ϵ equals 0, the DQL agent always selects the action with the highest Q-value, which can lead to local optima or sub-optimal decision making. With a non-zero ϵ, the DQL starts to explore unvisited S-A pairs and avoid local optima. However, aggressively increasing ϵ will result in an unstable control policy where DQL agents frequently take actions randomly instead of selecting the most suitable actions.

5.5 Overhead Analysis

We evaluate the overheads of the proposed SecureNoC design in the following aspects: timing, chip area, and power. Specifically, the timing overhead is obtained by GEM5, while the offline training time is not included. The chip area and static power consumption are evaluated using Synopsys Design Vision software with 32nm technology. Note that dynamic power consumption is application-specific, and it is not considered a portion of the power overhead.

Timing Overhead: The timing overhead is induced by D-ANN and DQL during the testing phase. The timing overhead is the latency used for output computation. In the worst case, the computation overheads of D-ANN and DQL are 60 and 100 cycles, respectively. To avoid the potential negative impact on network performance, we use two sets of different time steps for attribute monitoring and controlling. The two sets of time steps are offset by the ANN computation time which can pipeline the overhead effectively. By doing so, the calculation of ANNs does not block either the monitoring or the controlling. Therefore, the use of ANN will not negatively impact the overall performance.

Area and Power Overhead: In SecureNoC, for each router, the proposed bypass channel which is comprised of links and simple switches consumes 3350 μm^2 area. The proposed D-ANN and DQL consume additional 1364 μm^2 area for ALUs and 2290 μm^2 area for SRAM. Furthermore, the power overhead of the learning-based logics (D-ANN and DQL) is 0.33mw. For data encryption hardware, compared to the 4335 μm^2 area that the conventional AES consumes, the proposed on-demand light-weight data encryption method consumes 1169 μm^2 area. However, these overheads can be partially offset by replacing half of the router buffers with channel buffers, which implies 7485 μm^2 area reduction.

6 RELATED WORK

Researchers have proposed several techniques tackling HT attacks in NoCs [3]–[5], [10]–[14], [16]–[24], [43], [59]. For threat detection, built-in diagnosis hardware [5] periodically checks the correctness of the circuitry's logic operations and stalls the application execution during the diagnosis. FHL [3] and runtime monitoring [60], [61] dynamically monitor selected NoC attributes to capture abnormalities in NoC behaviors. For HT mitigation, SurfNoC [11] eliminates the transmissions between high-security domains by alternatively reserving and scheduling transmission channels in each dimension exclusively for a specific domain. NIBR [12] partitions the virtual channels of each router to transmit high-security data flows (that requires HT-free components) and low-security data flows (that allows HT-infected components) exclusively. In the data encryption aspect, conventional DES [26] and AES [27], [57] consist of complex computations and multiple encryption rounds to protect encrypted data from revealed by malicious attackers. However, the conventional DES and AES can be time-consuming and potentially vulnerable to side-channel attacks [43], [62]. To address these problems, simplified encryption methods [15] and trace masking techniques [43], [59], [63] are proposed, respectively. Fault tolerance designs (e.g. SECDED [64], [65], load-balancing [66], thermal aware routing [67], etc.) can mitigate the transient errors in the packet, yet the HTs can still generate excessive faults and incur retransmission traffic to congest the network. If the HTs are not detected and mitigated, the retransmission traffic can saturate network resources and induce network congestion, which leads to longer delays and excessive power consumption [68]–[70].

Machine learning techniques have been used in recent NoC designs to balance design trade-offs. These works tackle the challenges of enhancing network performance, improving NoC power, and mitigating errors [32], [35], [36], [52], [71]–[76]. SecureNoC differs from the previous designs by presenting accurate runtime HT detection, an efficient threat mitigation mechanism that fully utilizes network resources to improve network performance and power, and an on-demand light-weight data protection technique with minimized overhead.

7 Conclusions

We propose SecureNoC, a learning-enabled framework with architectural innovations and algorithm designs to enhance NoC security, performance, and power consumption holistically. We propose a learning-based threat detector in each router for accurate run-time HT detection. Using the detection results, the HT-infected routers are isolated, while packets with high security demands are propagated using the proposed multi-function bypass channels (MBCs) with reduced power and improved throughput. For data protection, we propose a lightweight dynamic encryption method that adapts to the diverse traffic patterns and security demands with minimized overhead. Additionally, we propose a deep-Q-learning (DQL)-based policy to handle the dynamic interactions and balance the trade-offs among the proposed techniques. Simulation results show that, as compared to current NoC security techniques, SecureNoC achieves 36% improved HT detection accuracy, reduces end-to-end packet latency by 39%, and decreases energy consumption by 46%. In this paper, we demonstrate the amplifying and synergistic effects of integrating architectural innovations with machine learning in a holistic approach to NoC design. In the future, we will expand the design space and focus on protecting the multicore system against broad attack models, which span hardware Trojan attacks, denial-of-service attacks, side-channel attacks, and other attacks on both on-chip and off-chip communications, with hardware innovations and algorithm enhancements.

Acknowledgment

This research was partially supported by NSF grants CCF-1702980, CCF-1901165, and CCF-1812495. We sincerely thank the anonymous reviewers for their excellent and constructive 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 inteconnection networks," in *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pp. 684–689, ACM, 2001.

[3] H. Salmani, "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, 2016.

[4] S. Charles, M. Logan, and P. Mishra, "Lightweight anonymous routing in NoC based SoCs," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE*, 2020.

[5] K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware trojan insertion," in *2013 IEEE international symposium on hardware-oriented security and trust (HOST)*, pp. 45–50, IEEE, 2013.

[6] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE design&test*, vol. 27, no. 1, pp. 10–25, 2010.

[7] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan horse detection using gate-level characterization," in *Proceedings of the 46th ACM/IEEE Design Automation Conference*, pp. 688–693, 2009.

[8] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[9] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric Trojans for fault-injection attacks on cryptographic hardware," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 18–28, IEEE, 2014.

[10] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE transactions on computers*, vol. 51, no. 5, pp. 541–552, 2002.

[11] H. M. Wassel, Y. Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, and T. Sherwood, "SurfNoC: a low latency and provably non-interfering approach to secure networks-on-chip," vol. 41, pp. 583–594, ACM New York, NY, USA, 2013.

[12] T. H. Boraten and A. K. Kodi, "Securing NoCs against timing attacks with non-interference based adaptive routing," in *Proceedings of the 12th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, IEEE, 2018.

[13] S. Sefton, T. Siddiqui, N. S. Amour, G. Stewart, and A. K. Kodi, "GARUDA: Designing energy-efficient hardware monitors from high-level policies for secure information flow," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2509–2518, 2018.

[14] F. Yao, G. Venkataramani, and M. Doroslovački, "Covert timing channels exploiting non-uniform memory access based architectures," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pp. 155–160, ACM, 2017.

[15] E. R. Naru, H. Saini, and M. Sharma, "A recent review on lightweight cryptography in iot," in *Proceedings of 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 887–890, IEEE, 2017.

[16] M. Hussain and H. Guo, "A bandwidth-aware authentication scheme for packet-integrity attack detection on Trojan infected NoC," in *Proceedings of 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 201–206, IEEE, 2018.

[17] Y. Lyu and P. Mishra, "Automated test generation for trojan detection using delay-based side channel analysis," in *Proceedings of 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1031–1036, IEEE, 2020.

[18] M. Hussain, A. Malekpour, H. Guo, and S. Parameswaran, "EETD: An energy efficient design for runtime hardware trojan detection in untrusted network-on-chip," in *Proceedings of 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 345–350, IEEE, 2018.

[19] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of dos attacks in NoC based SoCs," in *Proceedings of 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1160–1165, IEEE, 2019.

[20] M. Hussain and H. Guo, "Packet leak detection on hardware-trojan infected NoCs for MPSOC systems," in *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, pp. 85–90, 2017.

[21] Z. Zhang and Q. Yu, "Invariance checking based Trojan detection method for three-dimensional integrated circuits," in *Proceedings of 2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2020.

[22] A. Malekpour, R. Ragel, A. Ignjatovic, and S. Parameswaran, "Trojanguard: Simple and effective hardware Trojan mitigation techniques for pipelined MPSoCs," in *Proceedings of the 54th Annual Design Automation Conference*, pp. 1–6, 2017.

[23] A. Malekpour, R. Ragel, T. Li, H. Javaid, A. Ignjatovic, and S. Parameswaran, "Hardware trojan mitigation in pipelined mpsocs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 25, no. 1, pp. 1–27, 2020.

[24] V. Y. Raparti and S. Pasricha, "Lightweight mitigation of hardware trojan attacks in NoC-based manycore computing," in *Proceedings*

of 2019 56th ACM/IEEE Design Automation Conference (DAC), pp. 1–6, IEEE, 2019.

[25] J. Chen and G. Venkataramani, "Cc-hunter: Uncovering covert timing channels on shared processor hardware," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 216–228, IEEE, 2014.

[26] D. Coppersmith, "The data encryption standard (des) and its strength against attacks," *IBM journal of research and development*, vol. 38, no. 3, pp. 243–250, 1994.

[27] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[28] N. A. Gunathilake, W. J. Buchanan, and R. Asif, "Next generation lightweight cryptography for smart IoT devices:: implementation, challenges and applications," in *Proceedings of 5th World Forum on Internet of Things (WF-IoT)*, pp. 707–710, IEEE, 2019.

[29] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, *et al.*, "The gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[30] F. Yao, M. Doroslovacki, and G. Venkataramani, "Are coherence protocol states vulnerable to information leakage?," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 168–179, IEEE, 2018.

[31] T. Boraten, D. DiTomaso, and A. K. Kodi, "Secure model checkers for network-on-chip (noc) architectures," in *Proceedings of International Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 45–50, IEEE, 2016.

[32] K. Wang, H. Zheng, and A. Louri, "TSA-NoC: Learning-based threat detection and mitigation for secure network-on-chip architecture," *IEEE Micro*, vol. 40, no. 5, pp. 56–63, 2020.

[33] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *Journal of cryptology*, vol. 26, no. 1, pp. 80–101, 2013.

[34] T. Kitahara and R. K. Montoye, "Tri state buffer circuit for dual power system," Nov. 30 1993. US Patent 5,266,849.

[35] K. Wang and A. Louri, "CURE: A high-performance, low-power, and reliable network-on-chip design using reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2125–2138, 2020.

[36] 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 *Proceedings of 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, IEEE, 2019.

[37] 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 Computer Architecture News*, vol. 36, no. 3, pp. 241–250, 2008.

[38] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. 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*, pp. 333–346, IEEE Computer Society, 2006.

[39] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed systems*, vol. 3, no. 2, pp. 194–205, 1992.

[40] A. Singh, *Load-balanced routing in interconnection networks*. PhD thesis, Stanford University, 2005.

[41] H. Zheng, K. Wang, and A. Louri, "A versatile and flexible chiplet-based system design for heterogeneous manycore architectures," in *Proceedings of 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[42] Y. Li, A. Louri, and A. Karanth, "Scaling Deep-learning Inference with Chiplet-based Architecture and Photonic Interconnects," in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2021.

[43] T. S. Messerges, "Securing the aes finalists against power analysis attacks," in *International Workshop on Fast Software Encryption*, pp. 150–164, Springer, 2000.

[44] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[45] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A new (k, n)-threshold secret sharing scheme and its extension," in *Proceedings of International Conference on Information Security*, pp. 455–470, Springer, 2008.

[46] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang, "A (t, n) multi-secret sharing scheme," *Applied Mathematics and Computation*, vol. 151, no. 2, pp. 483–490, 2004.

[47] K. P. Burnham and D. R. Anderson, "Practical use of the information-theoretic approach," in *Model selection and inference*, pp. 75–117, Springer, 1998.

[48] C. L. Corniaux and H. Ghodosi, "An entropy-based demonstration of the security of shamir's secret sharing scheme," in *Proceedings of 2014 International Conference on Information Science, Electronics and Electrical Engineering*, vol. 1, pp. 46–48, IEEE, 2014.

[49] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[51] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *Proceedings of 56th Design Automation Conference (DAC)*, 2019.

[52] K. Wang, A. Louri, A. Karanth, and R. 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.

[53] Y. Chen and A. Louri, "Learning-Based Quality Management for Approximate Communication in Network-on-Chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, pp. 3724–3735, Nov. 2020.

[54] H. Zheng, K. Wang, and A. Louri, "Adapt-NoC: A flexible network-on-chip design for heterogeneous manycore architectures," in *Proceedings of 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 723–735, 2021.

[55] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[56] Y. Bai, V. W. Lee, and E. 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, pp. 825–838, 2017.

[57] H. K. Kapoor, G. B. Rao, S. Arshi, and G. Trivedi, "A security framework for noc using authenticated encryption and session keys," *Circuits, Systems, and Signal Processing*, vol. 32, no. 6, pp. 2605–2622, 2013.

[58] C. Bienia and K. 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.

[59] C. H. Gebotys, "A table masking countermeasure for low-energy secure embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 740–753, 2006.

[60] A. M. Fiskiran and R. B. Lee, "Runtime execution monitoring (REM) to detect and prevent malicious code execution," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 452–457, IEEE, 2004.

[61] Y. M. Edery, N. I. Vered, and D. R. Kroll, "Malicious mobile code runtime monitoring system and methods," June 6 2006. US Patent 7,058,822.

[62] C. Clavier, J.-S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 252–263, Springer, 2000.

[63] F.-X. Standaert, E. Peeters, and J.-J. Quisquater, "On the masking countermeasure and higher-order power analysis attacks," in *Proceedings of International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, vol. 1, pp. 562–567, IEEE, 2005.

[64] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A reliable network for unreliable silicon," in *Proceedings of 46th ACM/EDAC/IEEE Annual Design Automation Conference*, DAC'09, pp. 812–817, 2009.

[65] Y. Chen and A. Louri, "An Approximate Communication Framework for Network-on-Chips," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, pp. 1434–1446, June 2020.

[66] D. DiTomaso, A. Kodi, and A. 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, pp. 320–331, 2014.

[67] S.-Y. Lin, T.-C. Yin, H.-Y. Wang, and A.-Y. Wu, "Traffic-and thermal-aware routing for throttled three-dimensional network-on-chip systems," in *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*, pp. 1–4, IEEE, 2011.

[68] Y. Chen and A. Louri, "An online quality management framework for approximate communication in network-on-chips," in *Proceed-*

ings of the ACM International Conference on Supercomputing, ICS '19, pp. 217–226, ACM, June 2019.

[69] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: A defect-tolerant cmp switch architecture," in *Proceedings of The 12th International Symposium on High-Performance Computer Architecture*, HPCA'06, pp. 5–16, 2006.

[70] M. Koibuchi, H. Matsutani, H. Amano, and T. M. Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip," in *Proceedings of 2nd ACM/IEEE International Symposium on Networks-on-Chip*, NOCS'08, pp. 13–22, 2008.

[71] H. Zheng, K. Wang, and A. Louri, "Adapt-noc: A flexible network-on-chip design for heterogeneous manycore architectures," in *Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 723–735, 2021.

[72] J. Yin, S. Sethumurugan, Y. Eckert, C. Patel, A. Smith, E. Morton, M. Oskin, N. E. Jerger, and G. H. Loh, "Experiences with ml-driven design: A NoC case study," in *Proceedings of 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 637–648, IEEE, 2020.

[73] T.-R. Lin, D. Penney, M. Pedram, and L. Chen, "A deep reinforcement learning framework for architectural exploration: A routerless NoC case study," in *Proceedings of 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 99–110, IEEE, 2020.

[74] 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 *Proceedings of the 20th International Symposium on High Performance Computer Architecture*, HPCA'14, pp. 308–319, 2014.

[75] Y. Li and A. Louri, "ALPHA: a Learning-enabled High-performance Network-on-Chip Router Design for Heterogeneous Manycore Architectures," *IEEE Transactions on Sustainable Computing*, no. 2, pp. 274–288, 2021.

[76] H. Zheng and A. Louri, "Agile: A learning-enabled power and performance-efficient network-on-chip design," *IEEE Transactions on Emerging Topics in Computing*, 2020.

Ke Wang received the B.S. degree in Electrical Engineering from Peking University in 2013, and the M.S. degree in Electrical Engineering from Worcester Polytechnic Institute in 2015. He is currently working toward the Ph.D. degree in Computer Engineering in the School of Engineering and Applied Science at the George Washington University. His research work focuses on high-performance, energy-efficient, fault-tolerant, and secure interconnection networks.

Hao Zheng received the B.S. degree in electrical engineering from Beijing Jiaotong University, Beijing, China, and the M.S. degree in electrical engineering from George Washington University, Washington, DC, USA, where he is currently pursuing the Ph.D. degree in computer engineering. His research interests are in the areas of computer architecture and parallel computing, with emphasis on interconnection networks, machine learning techniques for efficient computing, and energy-efficient manycore architecture designs.

Yuan Li received the B.S. degree in physics from the University of Science and Technology of China in 2010, and the M.S. degree in microelectronics from the University of Newcastle upon Tyne in 2011. He is currently working toward the Ph.D. degree in computer engineering at the George Washington University. His research interests include machine learning architectures, accelerator-rich heterogeneous systems, and emerging interconnect and memory technologies.

Dr. Ahmed Louri is the David and Marilyn Karlgaard Endowed Chair Professor of Electrical and Computer Engineering at the George Washington University, which he joined in August 2015. He is also the director of the High Performance Computing Architectures and Technologies Laboratory. Dr. Louri received the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles, California in 1988. From 1988 to 2015, he was a professor of Electrical and Computer Engineering at the University of Arizona. From 2010 to 2013, Dr. Louri served as a program director in the National Science Foundation's (NSF) Directorate for Computer and Information Science and Engineering. Dr. Louri conducts research in the broad area of computer architecture and parallel computing, and recently he has been concentrating on: energy-efficient, reliable, and high-performance many-core architectures; accelerator-rich reconfigurable heterogeneous architectures; machine learning techniques for efficient computing, memory, and interconnect systems; emerging interconnect technologies (photonic, wireless, RF, hybrid) for NoCs; future parallel computing models and architectures (including convolutional neural networks, deep neural networks, and approximate computing); and cloud-computing and data centers. He is the recipient of the 2020 IEEE Computer Society Edward J. McCluskey Technical Achievement Award, "for pioneering contributions to the solution of on-chip and off-chip communication problems for parallel computing and manycore architectures." Dr. Louri is a Fellow of the IEEE, and he is currently the Editor-in-Chief of the IEEE Transactions on Computers. More information can be found at https://hpcat.seas.gwu.edu/Director.html.