On Optimizing Inter- and Intra-chiplet Interconnection Topologies for Robust Multi-chiplet Systems

Xiaohang Wang, *Member, IEEE*, Miao Xu, Amit Kumar Singh, *Member, IEEE*, Yingtao Jiang and Mei Yang, *Member, IEEE*

***Abstract*—Inter- and intra-chiplet interconnection networks play a vital role in the operation of many core systems made of multiple chiplets. However, these networks are susceptible to faults caused by manufacturing defects and attacks resulting from the malicious insertion of hardware Trojans and backdoors. Unlike conventional fault-tolerant or countermeasure methods, this paper focuses on optimizing network robustness to withstand both faults and attacks, while considering the constraints of chiplet area and power budget. To achieve this, this paper first defines network robustness as a quantifiable measure based on various network parameters, after which an optimization problem is formulated to optimize the robustness of the network topology. To efficiently solve this problem, a reinforcement learning algorithm is proposed. Experimental results demonstrate that the proposed method is capable of generating inter- and intra-chiplet interconnection networks that are significantly more robust than existing topology generation methods. Specifically, the proposed method improves robustness over ButterDonut and Kite, respectively by an average of 10.88% and 14.06% under random faults and by 9.37% and 7.81% under targeted attacks. These experimental results confirm that the proposed method is capable of generating robust inter- and intra-chiplet interconnection networks that can withstand both faults and attacks. By optimizing the network topology's robustness, it provides a valuable contribution to the design and security of chiplet-based core systems.**

***Index Terms*—chiplet, robustness, topology optimization**

I. INTRODUCTION

MULTI-CHIPLET systems have emerged as a new design paradigm aimed at improving chip yield and reducing chip manufacturing costs. Two representative architectures for such systems are illustrated in Fig. 1: GPU-based (Fig. 1 (a)) and CPU-based (Fig. 1 (b)) multi-chiplet systems. Despite their advantages, these systems face two significant threats: (1) random faults that arise from manufacturing imperfections [1] and can disrupt the operational reliability of the system, and (2) threats of attacks associated with the use of chiplets from untrusted sources or the integration of hardware Trojans or backdoors [2]. To address these challenges, a variety of fault tolerance and security countermeasure methods have been proposed, primarily focusing on repairing the remaining systems through packet rerouting within the inter- and intra-chiplet networks [3, 4]. One major limitation of these approaches, however, lie in their inability to address a fundamental requirement: *ensuring the resilience of the network topology against both faults and attacks*. In this context, faults are analyzed using a specific model in [22], while attacks are exemplified by the DoS attack model [24].

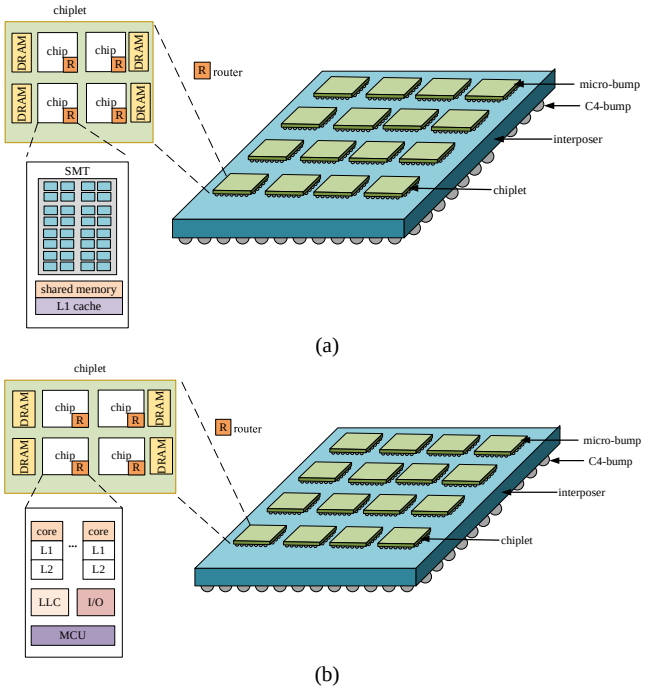


Fig. 1. Architecture models of the (a) GPU and (b) CPU based multi-chiplet systems.

A critical factor in ensuring the resilience of multi-chiplet system is network connectivity, which is evaluated at node or network levels using various parameters.

The degree of a node, which represents the number of direct connections (edges) it has in the network, serves as a key indicator of the network's connectivity. Increasing the degree

X. Wang is with State Key Laboratory of Blockchain and Data Security, Zhejiang University, China, and also with Hangzhou High-Tech Zone (Binjiang), Institute of Blockchain and Data Security (e-mail: xiaohangwang@zju.edu.cn). Xiaohang Wang is the corresponding author.

M. Xu is with the School of Software Engineering, South China University of Technology, China. (e-mail: sexumiao@mail.scut.edu.cn).

A. K. Singh is with the School of Computer Science and Electronic Engineering, University of Essex, UK. (e-mail: a.k.singh@essex.ac.uk)

Y. Jiang and M. Yang are with the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, USA. (e-mail: yingtao.jiang@unlv.edu, mei.yang@unlv.edu)

This work was supported in part by the National Natural Science Foundation of China under Grants 92373205 and 62374146, in part by the National Key Research and Development Program of China No. 2023YFB4404404, in part by the key R&D programme of Zhejiang Province No. 2024C01012, in part by the by Ant Group through CCF-Ant Research Fund, in part by the Key Technologies R&D Program of Jiangsu (Prospective and Key Technologies for Industry) under Grant BE2023005-2, and in part by CIE-Smartchip research fund No. 2023-004.

of a node enhances the network's robustness by providing additional connections, improving fault tolerance, and enabling alternative communication paths, thereby reducing the risk of network partitioning in the event of node or link failures. For example, consider the 4×4 mesh and torus topologies shown in Fig. 2. Suppose that both nodes 2 and 7 are faulty. In the mesh topology, these faulty nodes block all available paths for communication with node 3. In contrast, the torus topology retains an alternative path from node 1 to node 3 due to its additional wraparound connections. This example illustrates that topologies with higher degrees (such as the torus) exhibit greater robustness and fault tolerance compared to those with lower degrees (such as the mesh), even when subjected to the same set of node failures.

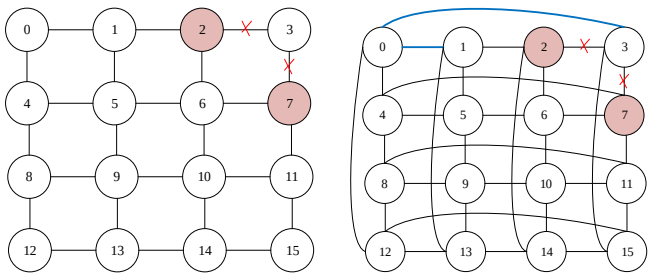
The robustness of a network is further quantified using the parameter γ, defined as the percentage of failed nodes required to disconnect the network ((indicated by infinite average network latency). A higher γ value signifies a more robust network. Fig. 3 shows a comparison of latency for different topologies under both attacks and faults. Under random faults, ButterDonut [5] exhibits a higher γ value than Kite [6]. However, under targeted attacks, Kite outperforms ButterDonut due to its greater number of high-degree nodes. This highlights the trade-offs between degree uniformity and robustness against specific failure types.

Based on the observation illustrated in Fig. 3, the paper proposes a methodology a novel methodology to model and optimize the robustness of inter- and intra-chiplet interconnection topologies. The key contributions of this work can be summarized as follows:

1) Inter- and intra-chiplet interconnection network robustness is modeled using structural parameters such as average neighbor degree, maximum degree, average path length, clustering coefficient, and the most frequent degree.

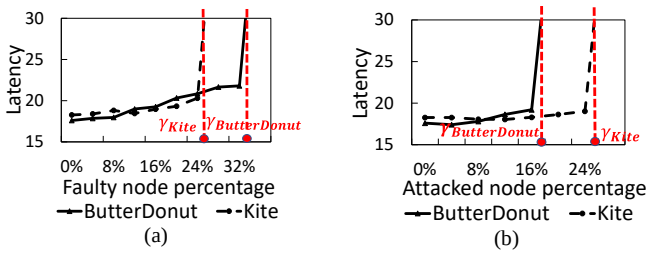
2) An optimization problem is formulated to optimize network robustness, while adhering to power and area constraints. To solve the problem, an efficient reinforcement learning algorithm is designed to find topologies with high robustness.

3) Experimental results demonstrate that the proposed method improves robustness over ButterDonut and Kite, respectively by an average of 10.88% and 14.06% under random faults and by 9.37% and 7.81% under targeted attacks.



(a) (b)

Fig. 2. (a) 4×4 mesh and (b) 4×4 torus.



(a) (b)

Fig. 3. The robustness of Kite and ButterDonut under (a) random faults and (b) targeted attacks.

The remainder of the paper is organized as follows. Section Ⅱ reviews the related work of chiplet-based many-core chips and their interconnection network topologies. Section Ⅲ models the robustness of a topology based on its network structural parameters. Section Ⅳ formally defines the problem of generating a robust inter- and intra-chiplet interconnection network topologies. Section Ⅴ details the reinforcement learning algorithm used to solve the optimization problem and generate robust topologies, and Section Ⅵ presents the experimental results, validating the proposed methodology. Finally, Section Ⅶ summarizes contributions and concludes the paper.

II. Related Work

In this section, we will first review the Network-on-Chip (NoC) topologies and survey the works concerning the generation of NoC topologies for many-core systems. These works aim to find efficient topologies that satisfy the specific requirements of the applications and systems. We will then examine the inter-chiplet network topologies, such as Kite, ButterDonut, and Hypercube, which enable efficient communication between chiplets. In the end, we will look into various techniques to mitigate the impact of faults and attacks, and the methods to improve the reliability and security of the interconnection network and the overall system through fault tolerance and security countermeasures.

A. NoC Topologies and Topology Generation

There are regular and irregular topologies in NoC designs. Among regular topologies, such as mesh [7], torus [8], butterfly [9], among a few others [10-15], mesh is particularly popular due to its simplicity and low cost. Torus improves connectivity and bandwidth but requires longer links.

Customized topologies are generated based on application communication characteristics, often by adding express long links for high-volume communications. These customized topologies serve different design goals, such as performance optimization and fault tolerance purposes.

For performance-optimization-oriented topologies, Katherine et al. [10] proposed a custom NoC topology generation method based on floorplanning, which achieves lower bandwidth and power consumption. Srinivasan et al. [11] used linear programming to generate application-specific NoCs, which reduces power consumption and router resources. Venkataraman et al. [12] used Ant Lion optimization techniques to generate topology with low power, small area, and high speed. Neeb et al. [13] used simulated annealing to map a task to a bidirectional chain topology and then add edges using a greedy algorithm, enabling scalability and expansibility.

For fault tolerance, Yang et al. [14] proposed the fault-tolerant cellular model, which adds a spare router in the center

of each hexagon to ensure high system reliability. Hosseinabady et al. [15] proposed de Bruijn graphs as on-chip interconnection networks and developed a routing algorithm that detours problematic links by at most two more switches further than the original route. However, these existing works do not especially consider network robustness and inter-chiplet interconnections.

B. Inter-chiplet Network Topologies and Fault Tolerance

In the literature, there are several studies on how to design efficient inter- and intra-chiplet interconnection networks. These works can be categorized into two main areas: (1) topology design for inter- and intra-chiplet networks, and (2) routing deadlock avoidance.

In the field of topology generation, Kite [6] allows for the use of longer links in Network-on-Interposer (NoI) and optimizes the effective hop count and effective bandwidth to improve communication throughput. Wang et al. [16] proposed a reusable NoI design for agile AI chip customization, which can self-adapt to the inter-die communication patterns of various neural network applications, enabling the reuse of the interposers for different applications. Li et al. [17] proposed a reusable general interposer architecture to amortize NRE costs and accelerate integration flows of interposers across different chiplet-based systems effectively. Sharma et al. [18] proposed a novel NoI architecture with multiple non-overlapping SFCs specifically targeting running multiple concurrent CNN inference tasks. Sharma et al. [19] developed an efficient multi-objective optimization algorithm to generate an NoP architecture where the number of links associated with each router (number of router ports) varies depending on the inter-chiplet traffic pattern.

For routing deadlock avoidance, Yin et al. [20] proposed a method to avoid inter-chiplet routing deadlock by introducing turn restrictions that break inter-chiplet cyclic dependencies. Taheri et al. [21] proposed a method to avoid inter-chiplet routing deadlock through a virtual network allocation strategy and improve fault tolerance through dynamic vertical link selection. Majumder et al. [22] proposed a selective injection control mechanism to prevent inter-chiplet routing deadlock. However, a common limitation of these works is the lack of consideration for network robustness in their designs. Robustness is crucial in ensuring the network's functionality and performance even in the presence of faults or attacks.

To combat faults/attacks in inter- or intra-chiplet networks, several approaches have been proposed, including (1) fault/attack detection, (2) fault-tolerant routing, and (3) security mechanisms. Previous studies [20, 21] have focused on detecting and locating faults and attacks in NoCs. These works aim to identify the presence of hardware Trojan attacks, Denial-of-Service (DoS) attacks and malicious traffic within the network. In [18, 22], fault-tolerant routing algorithms were introduced, such that once a fault/attack is detected, an alternative path is selected to bypass the faulty or malicious node. Regarding security mechanisms, works [3, 4] have proposed methods to enhance the defense capability against malicious nodes. These approaches involve monitoring network behavior or implementing authentication mechanisms to ensure the integrity and security of the communication.

Our work complements these existing approaches. We focus on generating a network topology that is inherently resilient to faults and attacks. By designing a robust topology, it can be integrated with online fault tolerance countermeasure schemes, enhancing the overall reliability and security of the inter- and intra-chiplet interconnection network. By combining the strengths of fault/attack detection, fault-tolerant routing, security mechanisms, and robust topology generation, it is possible to create a comprehensive framework that ensures the efficient and secure operation of inter- and intra-chiplet networks even in the presence of faults and attacks.

III. Modeling Robustness of Network Topology

In this section, we begin by introducing the robustness metric γ for a given network topology. This metric serves as a measure of the network's resilience and ability to withstand attacks or failures. Next, we delve into the modeling of γ in relation to network structure parameters. These parameters capture various aspects of the network's architecture, such as node connectivity, edge weights, or degree distributions.

A. Network Robustness Metric

Faults can be classified into two types: transient fault and permanent fault [22]. The former is usually caused by power grid fluctuations, particle hits and crosstalk, while permanent fault is caused by physical damages. For the threat model against chips, we consider two hardware trojan-assisted DoS attacks, namely sinkhole and blackhole attacks [24]. Blackhole attack disables links or router ports, while a sinkhole attack aggregates the traffic and intercepts the packets. The robustness (γ) of a network is defined as follows.

Definition 1. Robustness. Given a network $G(V, E)$ under random faults or targeted attacks, where V is the set of vertices and E is the set of edges, γ is the minimum percentage of faulty nodes or attacked nodes that cause the network to be disconnected, resulting in an infinite average latency.

Fig. 3 shows that γ is measured as 22.2% and 27.8% for an 8×8 torus under random faults and targeted attacks, respectively. As mentioned in [26], network robustness is closely related to network structure. Therefore, the robustness metric γ can be modeled statistically using various network parameters, including average path length (*APL*), average clustering coefficient (*ACC*), average neighbor degree (*AND*), the most frequent degree value of topology (k_{most}) and the highest degree value (k_{max}). By changing each network structure parameter, the correlation between the γ and each parameter is shown in Table I. One can see from Table I that *APL* is negatively correlated to γ, while *AND*, *ACC*, k_{max} and k_{most} display positive correlation with γ. Consequently, γ can be modeled by

$$\gamma = \alpha_6 * \frac{\alpha_1 * AND + \alpha_2 * ACC + \alpha_3 * k_{most} + \alpha_4 * k_{max}}{\alpha_5 * APL} + \alpha_7 \quad (1)$$

Eqn. (1) is derived empirically. The correlation between γ and each parameter is analyzed and summarized in Table I, which guides the design of the regression model in Eqn. (1). A maximum likelihood method [29] is used to determine the coefficients of this model.

To illustrate the metrics used in Eqn. (1), consider the 4×4 mesh in Fig. 4. These metrics, which characterize different aspects of the network topology, are computed as follows:

1. average neighbor degree (*AND*)

Definition: the sum of the degree values of all nodes, divided by the total number of nodes.

Example: In a 4×4 mesh with 16 nodes, there are 4 nodes with a degree of 2, 8 nodes with a degree of 3, and 4 nodes with a degree of 4. The total of degrees is (4×2) +(8×3) +(4×4) =48. Therefore, *AND*= 48/16 = 3.

2. average path length *(APL)*

Definition: the sum of the path lengths for all pairs of nodes, divided by the total number of node pairs.

Example: In the same 4×4 mesh, the distance between node $u_{i,j}$ and node $u_{m,n}$ is |*m*-*i*|+|*n*-*j*|, where *i*, *j*, *m*, *n* = 0, 1, 2… 15. the sum of the all pairwise distances is 480 and there are 16×15/2=120 node pairs. Therefore, *APL* = 480/120=4.

3. average clustering coefficient (*ACC*)

Definition: the sum of the clustering coefficients of all nodes, divided by the total number of nodes.

The clustering coefficient of a node $u_{i,j}$ defined as:

$$c_{i,j} = \frac{m_{i,j}}{\frac{k_{i,j}(k_{i,j}-1)}{2}}$$

where (i) $m_{i,j}$ is the number of actual direct edges between the neighbors of node $u_{i,j}$; (ii) $k_{i,j}$ is the degree of node $u_{i,j}$; and (iii) $\frac{k_{i,j}(k_{i,j}-1)}{2}$ gives the maximum number of possible direct connections among the neighbors of node $u_{i,j}$.

Example: For node $u_{1,0}$, the neighbors are nodes $u_{0,0}$, $u_{1,1}$, and $u_{2,0}$, but none of these neighbors are directly connected to each other. Hence, $m_{1,0} = 0$. Although $k_{1,0} = 3$, node $u_{1,0}$'s clustering coefficient is $\frac{0}{\frac{3(3-1)}{2}} = 0$. In this mesh, actually every node has a clustering coefficient of 0, so, the *ACC* = 0.

4. the maximum degree (k_{max})

Definition: the maximum number of direct connections (edges) that any single node in the network has.

Example: In the 4×4 mesh, the maximum degree is 4. For instance, nodes 5, 6, 9, and 10, each has 4 connections.

5. the most frequent degree value (k_{most})

Definition: the degree value with the highest frequency.

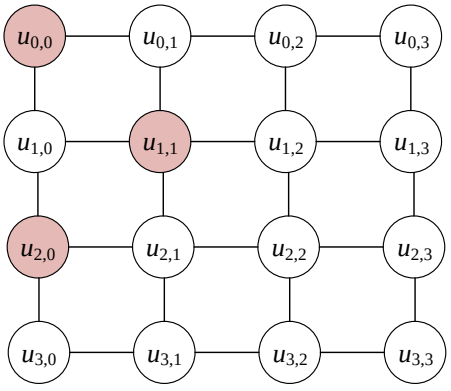


Fig. 4. An example showing the computation of the network parameters.

Example: In the 4×4 mesh, the most frequent degree value is 3, since 8 out of 16 nodes have a degree value of 3, while 4 nodes have degrees of 2 or 4.

B. Modeling γ in Multi-chiplet Interconnection Networks

In Fig. 5 (a), the network topology is defined as a graph $G(V, E)$, where each vertex $u_{i,j} \in V$ is tile i in chiplet j, and $e_{i,k,j} \in E$ is the link between $u_{i,j}$ and $u_{k,j}$ in chiplet j. $e_{i,k,j} = 1$ if there is a link between $u_{i,j}$ and $u_{k,j}$, and 0 otherwise. The chiplets in the system are denoted by $R_1, \cdots, R_c$, where each $R_j \subset V, j = 1, \cdots, c$, is a set of tiles in chiplet j. The number of tiles in R_j is given as $|R_j|$, and c is the total number of chiplets in the system. In the context of using D2D (die to die) interfaces within each chiplet, as described in [21], it is assumed that each chiplet has no more than four D2D interfaces for inter-chiplet communication [21]. The count of D2D interfaces in chiplet j is denoted as d_j, and the D2D interfaces in the chiplet j are indexed by nodes $u_{|R_j|+1,j}, \ldots, u_{|R_j|+d_j,j}$ within chiplet j. Therefore, the total number of nodes in chiplet j is $|R_j| + d_j$. Given $|V|$, c and $|R_j|$ should hold the following:

$$\sum_{j=1}^{c} |R_j| = |V| \qquad (2)$$

In Fig. 5 (a), the D2D interfaces within a chiplet can connect to any tile within the same chiplet. This connectivity is represented as $e_{i,|R_j|+m,j} \in E$, where i is a tile within chiplet j, and m is the index of the D2D interface within the same

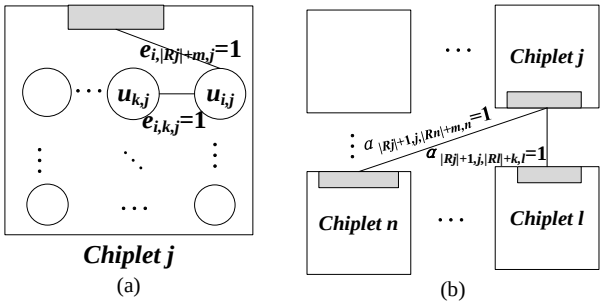


Fig. 5. An example of the variables describing the (a) intra-chiplet connection and (b) inter-chiplet connection.

TABLE I. CORRELATIONS BETWEEN ϒ AND NETWORK PARAMETERS

Network	*AND*	*APL*	*ACC*	k_{most}	k_{max}
Random faults					
Mesh	0.781	-0.865	0	0.772	0
Torus	0.945	-0.999	0	0.539	0
Kite	0.890	-0.998	0.665	0.506	0.276
Butterfly	0.853	-0.962	0.243	0.253	0.392
Targeted attacks					
Mesh	0.909	-0.992	0	0.692	0
Torus	0.880	-0.963	0	0.720	0
Kite	0.728	-1	0.556	0.593	0
Butterfly	0.783	-0.986	0.342	0.435	0.228

chiplet. In Fig. 5 (b), the link between D2D interfaces in chiplet j and chiplet n is denoted by $\alpha_{|R_j|+i,j,|R_n|+m,n}$. If there is a link between $u_{|R_j|+i,j}$ and $u_{|R_n|+m,n}$, the value of $\alpha_{|R_j|+i,j,|R_n|+m,n}$ is 1; otherwise, it is 0.

1) Computing AND

As per [28], the average neighbor degree is calculated using (3). *AND* is the average degree value of each node in the network, which can be obtained by summing up the degree values of all nodes and dividing it by the total number of nodes in the network. In this case, the total number of nodes in the network is given as the tile count plus the number of interface nodes ($|V| + \sum_{j=1}^{c} d_j$). The degree value of a node is the number of edges connected to that node. Note that the sum of all nodes is equal to twice the number of edges in the network, encompassing both the edges connecting the tile nodes and the edges connecting the interface nodes.

$$AND = \frac{\sum_{i=1}^{|R_j|}\sum_{k=1}^{|R_j|+d_j}\sum_{j=1}^{c} e_{i,k,j} + \sum_{p=|R_j|+1}^{|R_j|+d_j}\sum_{j=1}^{c}\sum_{q=|R_n|+1}^{|R_n|+d_n}\sum_{n=1}^{c}\alpha_{p,j,q,n}}{|V| + \sum_{j=1}^{c} d_j} \quad (3)$$

2) Computing APL

As per [28], the average shortest path length is computed using (4),

$$APL = \frac{1}{(|V| + \sum_{j=1}^{c} d_j)(|V| + \sum_{j=1}^{c} d_j - 1)}\left(\sum_{i=1}^{|R_j|+d_j}\sum_{j=1}^{c}\sum_{m=1}^{|R_n|+d_n}\sum_{n=1}^{c} l_{i,j,m,n}\right) \quad (4)$$

This equation calculates the average path length between any two nodes in the network, which requires the computation of the shortest path length $l_{i,j,m,n}$ of any two nodes. For nodes within the same chiplet, the shortest path length is computed using (5),

$$l_{i,j,m,n} = min\left(e_{i,m,j}, e_{i,a,j} + l_{a,m,j}\right)$$
$$\forall 0 < a, i, m \le |R_j| + d_j, 1 \le j, n \le c, j = n, e_{i,m,j}, e_{i,a,j} \ne 0 \quad (5)$$

Take Fig. 6 (a) as an example. As nodes $u_{i,j}$ and $u_{b,j}$ in chiplet j are not directly connected, the shortest path length from $u_{i,j}$ to $u_{b,j}$ is equal to the minimum path length from $u_{i,j}$ to other nodes, plus the shortest path length from other nodes to $u_{b,j}$. This is caluculated as $\min\left\{e_{i,i+1,j} + l_{i+1,b,j}, e_{i,i+2,j} + l_{i+2,b,j}, \dots, e_{i,|R_j|+d,j} + l_{|R_j|+d,b,j}\right\}$.

For nodes in different chiplets, the shortest path is calculated by finding the shortest path from the source node to the D2D interface within the same chiplet using (5), plus the shortest distance from the D2D to the destination node. The shortest path length for nodes in different chiplets is thus calculated using (6),

$$l_{i,j,m,n} = \min\left(l_{i,j,|R_j|+k,j} + l_{|R_j|+k,j,|R_n|+h,n} + l_{|R_n|+h,n,m,n}\right)$$
$$l_{|R_j|+k,j,|R_n|+h,n} = \min\left(\alpha_{|R_j|+k,j,|R_n|+h,n}, \alpha_{|R_j|+k,j,|R_a|+b,a} + l_{|R_a|+b,a,|R_n|+h,n}\right)$$
$$\forall 0 < i \le |R_j|, 0 < m \le |R_n|, 0 < n, j, a \le c, 0 < k \le d_j,$$
$$0 < h \le d_n, 0 < b \le d_a, j \ne n \quad (6)$$

As shown in Fig. 6 (b), the shortest path from node $u_{i,j}$ in chiplet j to node $u_{m,n}$ in chiplet n consists of three parts: 1) the shortest path from source node $u_{i,j}$ to D2D in chiplet j, 2) the shortest path from chiplet j to chiplet n, and 3) the shortest path from D2D to the destination node $u_{m,n}$ in chiplet n.

3) Computing ACC

According to [28], *ACC* is computed using (7) – (9) as follows:

$$ACC = \frac{1}{|V| + \sum_{j=1}^{c} d_j}\left(\sum_{i=1}^{|R_j|+d_j}\sum_{j=1}^{c}\frac{2m_{i,j}}{k_{i,j}(k_{i,j}-1)}\right) \quad (7)$$

$$m_{i,j} = \sum_{a=1}^{|R_j|+d_j}\sum_{b=1}^{|R_j|+d_j} e_{i,a,j} * e_{i,b,j} * e_{a,b,j} + \sum_{p=|R_q|+1}^{|R_q|+d_q}\sum_{q=1}^{c}\sum_{m=|R|+1}^{|R_n|+d_n}\sum_{n=1}^{c}\alpha_{i,j,m,n} * \alpha_{i,j,p,q} * \alpha_{p,q,m,n} \quad (8)$$

$$k_{i,j} = \sum_{k=1}^{|R_j|+d_j} e_{i,k,j} + \sum_{m=|R_n|+1}^{|R_n|+d_n}\sum_{n=1}^{c}\alpha_{i,j,m,n} \quad (9)$$

where $m_{i,j}$ is the sum of connected edges between node $u_{i,j}$'s neighbors, and $k_{i,j}$ is the degree of node $u_{i,j}$. In simpler terms, the clustering coefficient of a node is the ratio of actual connected edges ($m_{i,j}$) between the node's neighbors to the maximum possible connected edges ($k_{i,j} * (k_{i,j} - 1)/2$). If $u_{i,j}$ has neighbors $u_{a,j}$ and $u_{b,j}$ and $u_{a,j}$ and $u_{b,j}$ are also neighbors to each other, the product of$e_{i,j,a,j}$, $e_{i,j,bj}$, $e_{a,j,b,j}$ is 1; otherwise, it is 0.

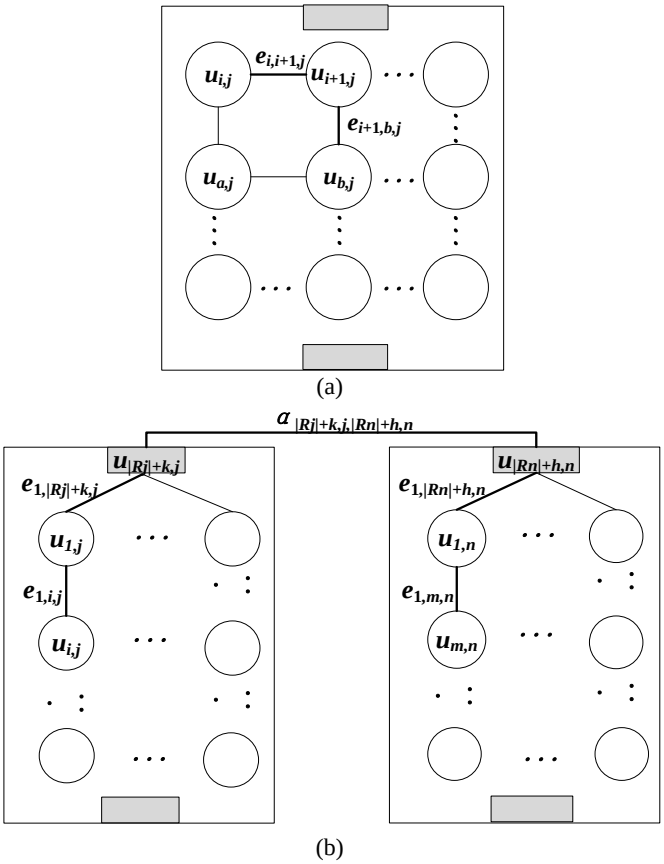


Fig. 6. An example showing the calculation of *APL* in two scenarios: (a) two nodes are within the same chiplet, and (b) two nodes are in different chiplets.

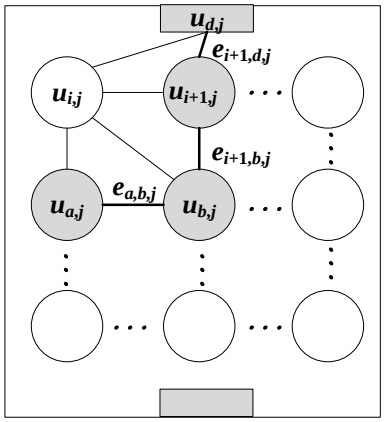


Fig. 7. The diagram of computing *ACC*.

For example, as shown in Fig. 7, if node $u_{i,j}$ has 4 neighbor nodes $(u_{d,j}, u_{i+1,j}, u_{a,j}, u_{b,j})$, with $k_{i,j} = 4$, the maximum number of connected edges between neighbors is $\frac{k_{i,j}(k_{i,j}-1)}{2} = 6$. If the actual connected edges between neighbors are 3 (i.e., $e_{i+1,d,j}$, $e_{i+1,b,j}$, $e_{a,b,j}$), then $m_{i,j} = 3$. Therefore, the clustering coefficient of node $u_{i,j}$ is $\frac{2m_{i,j}}{k_{i,j}(k_{i,j}-1)} = 1/2$.

4) Computing k_{most}

The highest degree frequency is calculated as in (10) by using Pearson's empirical coefficient [27]:

$$k_{most} = AND - 3\left(AND - \frac{k_{max} - k_{min}}{2}\right) \quad (10)$$

$$k_{min} = min(k_{i,j}), \forall\, 0 < i \le |R_j| + d_j, 0 < j \le c \quad (11)$$

$$k_{max} = max(k_{i,j}), \forall\, 0 < i \le |R_j| + d_j, 0 < j \le c \quad (12)$$

IV. Problem Definition

The problem of optimizing the robustness of inter- and intra-chiplet interconnection network topology is formulated as follows. Given the network topology within a multi-chiplet system, the objective is to identify the inter- and intra-network configurations that maximizes the robustness metric γ while adhering to power and area constraints.

Mathematically, we can formulate this problem as follows:

$$\max\ \gamma = \alpha_6 * \frac{\alpha_1 * AND + \alpha_2 * ACC + \alpha_3 * k_{most} + \alpha_4 * k_{max}}{\alpha_5 * APL} + \alpha_7 \quad (13)$$

$$s.t. \quad A(R_i) \le A_0 \quad \forall\, i = 1, \cdots, c \quad (14)$$

$$P(R_i) \le P_0 \quad \forall\, i = 1, \cdots, c \quad (15)$$

$$e_{i,k,j} = e_{k,i,j} = 0\ or\ 1, \forall\, 0 < i, k < |R_j| + d_j, 0 < j \le c \quad (16)$$

$$\alpha_{ijmn} = \alpha_{mnij} = 0\ or\ 1,$$
$$\forall |R_j| + 1 \le i \le |R_j| + d_j, |R_n| + 1 \le m \le |R_n| + d_n, 0 < j, n \le c \quad (17)$$

$$e_{i,k,j} = 0, \forall\, i = k \,||\, |R_j| + 1 \le i, k \le |R_j| + d_j \quad (18)$$

where $A(R_i)$ and $P(R_i)$ are the area and power of chiplet R_i, and A_0 and P_0 are the area and power thresholds of a chiplet.

Equation (18) specifies that a tile can not connect to itself. For passive interposers, where only adjacent chiplets can be connected, we introduce additional constraints:

$$\alpha_{i,j,m,n} = 0\ or\ 1, \forall\, |R_j| + 1 \le i \le |R_j| + d_j,$$
$$|R_n| + 1 \le m \le |R_n| + d_n, n = j + 1\ or\ j + \lceil\sqrt{c}\rceil, 0 < j, n \le c \quad (19)$$

$$\alpha_{i,j,m,n} = 0, \forall\, |R_j| + 1 \le i \le |R_j| + d_j,$$
$$|R_n| + 1 \le m \le |R_n| + d_n, \forall j \% \lceil\sqrt{c}\rceil = 0, n = j + 1 \quad (20)$$

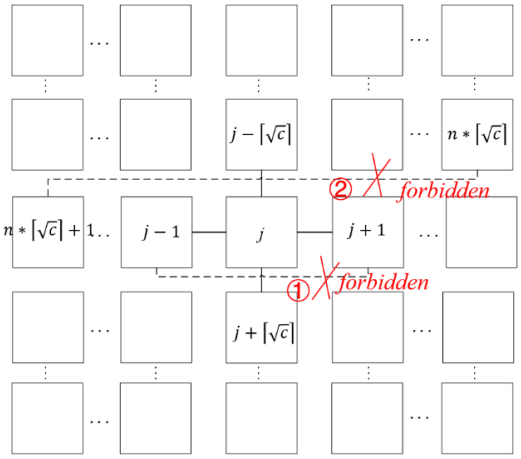


Fig. 8. Illustration of (19) and (20).

As illustrated in Fig. 8, chiplet j is restricted to establishing connections only with the chiplets $j \pm 1$ and $j \pm \lceil\sqrt{c}\rceil$. In other words, chiplet $j - 1$ cannot establish a connection with chiplet $j + 1$ (marked as ②). In a specific scenario, as shown in Fig. 8 (marked as ①), the edge chiplets are subject to a constraint where they cannot connect with the chiplets whose numbers are given by $n * \lceil\sqrt{c}\rceil$ (n=0, 1, …, $\lceil\sqrt{c}\rceil$). In simple terms, the edge chipet labeled as $n * \lceil\sqrt{c}\rceil$ cannot connect with the chiplet labeled as $n * \lceil\sqrt{c}\rceil + 1$.

The area of a chiplet R_j is computed as follows, where $A(u_{i,j})$ is the area model that returns the area of each router (i from 1 to $|R_j|$) and $A(u_{i,j})$ returns the area of each D2D interface (i from $|R_j| + 1$ to $|R_j| + d_j$).

$$A(R_j) = \sum_{i=1}^{|R_j|} A(u_{i,j}) + \sum_{i=|R_j|+1}^{|R_j|+d_j} A(u_{i,j})\ \forall\, 0 < j \le c \quad (21)$$

Similarly, the power of a chiplet R_j is computed as follows.

$$P(R_j) = \sum_{i=1}^{|R_j|} P(u_{i,j}) + \sum_{i=|R_j|+1}^{|R_j|+d_j} P(u_{i,j}), \forall\, 0 < j \le c \quad (22)$$

where $P(u_{i,j})$ is the power model that returns the power of each router (i from 1 to $|R_j|$) and $P(u_{i,j})$ returns the power of each D2D interface (i from $|R_j| + 1$ to $|R_j| + d_j$).

V. Topology Generation Algorithm

To solve the above problem, a reinforcement learning based algorithm is proposed in this section. The algorithm works on a search tree iteratively, with the root node being a fully connected topology as the initial solution, and each node branches new child node through actions selected by the UCD algorithm [51]. The algorithm disconnects the links iteratively to achieve the maximum robustness under area and power constraints.

A. Initial Solution

We use the state $\boldsymbol{s}$ to represent the inter- and intra-chiplet interconnection network topology, and the state in the

algorithm as an $N \times N$ adjacent matrix (that is, the inter- and intra-chiplet network topology), where $N = \sum_{j=1}^{c} (|R_j| + d_j)$, with $\sum_{j=1}^{c} d_j$ being the number of D2D and $\sum_{i=1}^{c} |R_j|$ being the number of tiles in the multi-chiplet system. Fig. 9 (a) shows the adjacency matrix of an inter- and intra-chiplet interconnection network topology. One can see from Eqn. (1) that *APL* is negatively correlated to γ, while *AND*, *ACC*, k_{max}, and k_{most} have positive correlation with γ. The *APL* value of the fully connected network is the smallest, while *AND*, *ACC*, k_{max}, and k_{most} are the largest. Therefore, the topology with highest robustness without considering the area and power constraints corresponds to a fully connected network. We set the initial solution $\boldsymbol{s_0}$ to be a matrix whose elements are all 1.

B. Reward Function in the Reinforcement Learning

An action in the reinforcement algorithm is a quadruple (x, y, m, n), which corresponds to deleting the link between nodes $u_{x,y}$ and $u_{m,n}$. The reward for moving from state $\boldsymbol{s_1}$ to state $\boldsymbol{s_2}$ is

$$R(\boldsymbol{s_1}, \boldsymbol{s_2}) = \gamma(\boldsymbol{s_2}) - \gamma(\boldsymbol{s_1}) + c_1\big(A(\boldsymbol{s_1}) - A(\boldsymbol{s_2})\big) + c_2\big(P(\boldsymbol{s_1}) - P(\boldsymbol{s_2})\big) \tag{23}$$

where $\gamma(\boldsymbol{s_i})$ is the robustness of state $\boldsymbol{s_i}$, $A(\boldsymbol{s_i})$ is the area of state $\boldsymbol{s_i}$, $P(\boldsymbol{s_i})$ is the power of state $\boldsymbol{s_i}$. The coefficients c_1 and c_2 are used to bias the weights of robustness, area, and power in optimization.

C. Strategy and Value Network

Since the inter- and intra-chiplet network is modeled as a graph, in order to better converge along the search direction of the UCD algorithm, this paper uses ResGCN model [53] as the strategy and value network (denoted as the SV model).

Fig. 9 (c) shows the entire strategy and value network of the topology generation method composed by multiple GCN and ResGCN blocks, and Fig. 9 (b) shows the ResGCN module where H is the number of channels. The model fits both the strategy and the value functions. The convolution layer is followed by a batch normalization layer, and some convolution layers (highlighted in Fig. 9 (c)) are followed by a respective TopK pooling layer. The equation for each building block of ResGCN is as follows:

$$\boldsymbol{\mathcal{G}}_{l+1} = \mathcal{F}(\boldsymbol{\mathcal{G}}_l, \boldsymbol{\mathcal{W}}_l) + \boldsymbol{\mathcal{G}}_l \tag{24}$$

where $\boldsymbol{\mathcal{G}}_l$ is the input of the l-th layer, $\boldsymbol{\mathcal{W}}_l$ is the learnable parameter of the l-th layer, and $\mathcal{F}(\boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{W}})$ is the residual mapping. The final output strategy needs to be passed through Rectified Linear Unit (ReLU) [54] and normalized exponential function (softmax function) [55], where a $4 \times N$ matrix $\boldsymbol{P} = (p_{i,j})$ output is obtained, with $p_{1,\beta}, p_{2,\delta}, p_{3,\sigma}, p_{4,\tau}$ being the probabilities of $x = \beta, y = \delta, m = \sigma, n = \tau$ respectively and $\sum_{\beta=1}^{N} p_{1,\beta} = 1, \sum_{\delta=1}^{N} p_{2,\delta} = 1, \sum_{\sigma=1}^{N} p_{3,\sigma} = 1, \sum_{\tau=1}^{N} p_{4,\tau} = 1$.

The SV model is defined as $(\boldsymbol{P}, q) = g_{\boldsymbol{\theta}}(\boldsymbol{s})$, where $\boldsymbol{P}$ is the predicted action probabilities, q is the predicted reward. The training data is a triplet $(\boldsymbol{s}, \boldsymbol{\pi}, r)$, where $\boldsymbol{s}$ is a state, $\boldsymbol{\pi}$ is the vector of action probabilities, and r is state reward. The loss function of the SV model is

$$l = (r - q)^2 - \langle \boldsymbol{\pi}, \log \boldsymbol{P} \rangle + b\|\theta\|^2 \tag{25}$$

where b is the regularization parameter, $\langle \boldsymbol{\pi}, \log \boldsymbol{P} \rangle$ are the inner product of $\boldsymbol{\pi}$ and $\log \boldsymbol{P}$. In this paper, the SV model uses the Adam algorithm [52] for parameter optimization.

D. Reinforcement Learning Algorithm

Given a search tree $T(S, G)$, where $\boldsymbol{s_i} \in S$ is the search node, $\boldsymbol{s_i}$ is a state (*i.e.*, an inter- and intra-chiplet interconnection network topology), and $g_{i,j} = (\boldsymbol{s_i}, \boldsymbol{s_j})$ is the edge between $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$, $g_{i,j}$ is an action of deleting a link in the topology. The UCD algorithm is used to search an action. $n(g_{i,j})$ and $\mu(g_{i,j})$ are visit number and average reward value of edge $g_{i,j}$ in the search tree, respectively. The set of outgoing edges from the destination node of edge $g_{i,j}$ is denoted as $O(\boldsymbol{s_j})$, and the set of outgoing edges from the source node of edge $g_{i,j}$ is denoted as $O(\boldsymbol{s_i})$. As Fig. 10 shows, during iteration 5, state $\boldsymbol{s_0}$ selects action a_5 and results in state $\boldsymbol{s_2}$ (in iteration 2, $\boldsymbol{s_0}$ selects action a_2 and it also reaches $\boldsymbol{s_2}$). At this point, the current state is $\boldsymbol{s_2}$, the set $O(\boldsymbol{s_0})$ is $\{g_{0,1}, g_{0,2}, g_{0,4}\}$ and set $O(\boldsymbol{s_2})$ is $\{g_{2,3}\}$. The visit number and average reward value of edge $g_{i,j}$ when $O(\boldsymbol{s_j}) = \emptyset$ are denoted as $n'(g_{i,j})$ and $\mu'(g_{i,j})$, respectively, and d is the depth of backtracking. According to [51], in each iteration, the visit number $(n_d(g_{i,j}))$, average reward $(\mu_d(g_{i,j}))$, and parent node's visit number $(p_d(g_{i,j}))$ of edge $g_{i,j}$ are updated as follows:

$$n_d(g_{i,j}) = \begin{cases} n(g_{i,j}), d = 0 \\ \sum_{f \in O(\boldsymbol{s_j})} n_{d-1}(f), d > 0 \end{cases} \tag{26}$$

$$\mu_d(g_{i,j}) = \begin{cases} \mu(e), d = 0 \\ \dfrac{\mu'(g_{i,j}) n'(g_{i,j}) + \sum_{f \in O(\boldsymbol{s_j})} \mu_{d-1}(f) n(f)}{n'(g_{i,j}) + \sum_{f \in O(\boldsymbol{s_j})} n(f)}, d > 0 \end{cases} \tag{27}$$

$$p_d(g_{i,j}) = \sum_{f \in O(\boldsymbol{s_i})} n_d(f) \tag{28}$$

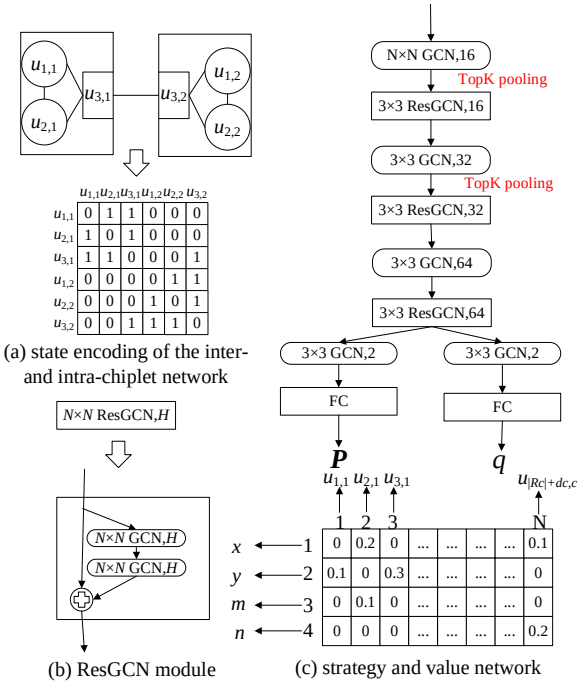


Fig. 9. The strategy and value network.

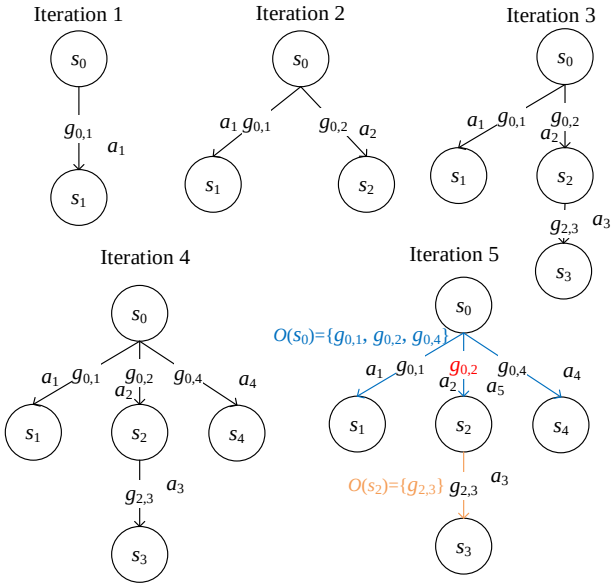


Fig. 10. An example of the search tree.

According to the UCD algorithm [51], the upper confidence bound of edge $g_{i,j}$ ($u_{d_1,d_2,d_3}(g_{i,j},\rho)$) is as follows:

$$u_{d_1,d_2,d_3}(g_{i,j},\rho)=\mu_{d_1}(g_{i,j})+\theta\rho\sqrt{\frac{\log p_{d_2}(g_{i,j})}{n_{d_3}(g_{i,j})}} \quad (29)$$

where θ is a constant, d_1, d_2, d_3 are 1, 2, 3, which are tracing depths of 1, 2, 3, respectively. A larger θ makes the algorithm more inclined to explore new branches. ρ is the probability of selecting the action corresponding to edge $g_{i,j}$ given by the SV model. It can be seen that the selected action in state $\boldsymbol{s}$ is:

$$a^*=\underset{a}{\operatorname{argmax}}\ u_{d_1,d_2,d_3}\big(g(\boldsymbol{s},a),\boldsymbol{M}(\boldsymbol{s})*\boldsymbol{P}(\boldsymbol{s},a)\big) \quad (30)$$

where $g(\boldsymbol{s},a)$ is the edge corresponding to selecting action a in state $\boldsymbol{s}$, $\boldsymbol{P}(\boldsymbol{s},a)$ is the probability of selecting action a in state $\boldsymbol{s}$ given by the SV model, and $\boldsymbol{M}(\boldsymbol{s})$ is the adjacency matrix of state $\boldsymbol{s}$.

In summary, the topology generation algorithm includes the following steps in each iteration:

1) *Selection*: Starting from the root node, actions are selected by using (30) until the current node is a leaf node. Node generation must be passed through a node hash table to avoid multiple nodes with the same state being created. It is performed as follows: the initial state is set to $\boldsymbol{s_0}$ and nodes are stored in the hash table ($hash\_table(\boldsymbol{s})$). The current state is set to $\boldsymbol{s}=\boldsymbol{s_0}$. If the current node is not a leaf node, the SV model is used to calculate the probability distribution of actions $\boldsymbol{P}(\boldsymbol{s},a)$ and reward $q(\boldsymbol{s})$, and an action is selected by using (30) to generate a new node.

2) *Expansion and evaluation*: If the current state $\boldsymbol{s}$ is a terminal state, the cumulative reward $R(\boldsymbol{s})$ is obtained by calculating the path reward from the root node to the current node using the reward function; otherwise, for the current node $\boldsymbol{s}$, children nodes corresponding to all feasible actions are created and the selection probability $\boldsymbol{P}(\boldsymbol{s},a)$ of each action and the reward $q(\boldsymbol{s})$ of the state $\boldsymbol{s}$ are predicted by using the SV model. The SV model is being trained on the results of search iterations.

3) *Backtracking*: Edges related to the selected nodes are updated. The set of nodes on the traversal path is denoted as D, and the ancestors of node $\boldsymbol{s_i}$ within distance d is defined as

$$\boldsymbol{\Pi_d}(\boldsymbol{s_j})=\begin{cases}\{\boldsymbol{s_i}\}, d=0\\ \{\boldsymbol{s_i}|g_{i,j}\in O(\boldsymbol{s_j})\}\cup\boldsymbol{\Pi_{d-1}}(\boldsymbol{s_i}), d>0\end{cases} \quad (31)$$

For edge $g_{i,j}\in\{g_{h,w}|h,w\in \mathrm{D}\cup\boldsymbol{\Pi_1}(\boldsymbol{s})\}$, visit number $n(g_{i,j})$ is updated to be $n(g_{i,j})+1$. For edge $g_{i,j}\in\{g_{h,w}|h,w\in \mathrm{D}\cup\boldsymbol{\Pi_2}(\boldsymbol{s})$, average award $\mu(g_{i,j})$ is updated to be $\mu(g_{i,j})+\Delta\mu(g_{i,j})$. For the incoming edges of the current node $\boldsymbol{s}$, $\Delta\mu(g_{i,j})=q(s)$; for other edges, $\Delta\mu(g_{i,j})=\frac{\sum_{f\in O(g_{i,j})}\Delta\mu(f)n(f)}{n'(g_{i,j})+\sum_{f\in O(g_{i,j})}n(f)}$.

4) The above steps are repeated until the iteration upper bound is reached, and the leaf node with the highest reward is output.

VI.A Experimental Setup

Our experiments were performed using the cycle-accurate multi-chiplet simulator [38] which can simulate both x86 and GPU multi-chiplet systems. The inter- and intra-chiplet network simulators can also run individually without cores with random traffic injection as input (*i.e.*, network only mode). In the GPU multi-chiplet system, each network node is a graphics processing cluster (GPC), each of which has multiple texture processing clusters (TPCs) containing multiple streaming multiprocessors (SMs) and L1 caches. Additionally, there is both a private and shared TLB hierarchy assisted by a hardware page table walker (PTW). In the x86 multi-chiplet system, multiple x86 many-core chiplets are connected by the inter-chiplet network. Table II lists the configurations of the multi-chiplet system simulators.

Our benchmark suit has both random traffics and real applications. The random traffics are generated following the parameters specified in Table II by running the simulation with the network only mode. Additionally, there are two types of real benchmarks. The first set of real benchmarks are PARSEC and SPLASH-2 running on the x86 multi-chiplet system simulator with the configurations summarized in Table II. The second real benchmarks run on the multi-chiplet systems, including ResNet [29], Transformer [30], GCN [31] and CNN [32] for the GPU multi-chiplet system. The datasets of ResNet and CNN are CIFAR-10 [33] and MNIST [34], respectively. The dataset of GCN and Transformer both are VOC [35]. The benchmarks of neural network adopt the model parallelization approach as follows [36]. Each chiplet is responsible for computing tasks at one or more layers of the model. In the chiplet, the matrix multiplication of the data sets of the neural network is decomposed into multiple vectors multiplied by the matrix and assigned to different cores [37]. Both the baseline inter- and intra-chiplet topologies are mesh. The network sizes are 4×9, 4×16, and 4×36, where the first term is the chiplet count and the second term is the tile count per chiplet.

The transmission latency is divided among three components: 1) packetization and depacketization times (the values are obtained from [61, 62]); 2) transceiver latency (the

values are obtained from [56, 57]); 3) interposer wire latency (values from [40]).

The base area and power parameters used in this study come from reliable sources and are validated by simulations. Processor element areas within each chiplet are obtained from [43] and McPAT [60] for GPU and CPU based multi-chiplets, respectively. The power consumption of GPU and CPU multi-chiplet systems is simulated using GPUWattch and McPAT, respectively. For routers and SRAM components, their area and power requirements are calculated using DSENT [46] and CACTI 6.0 [47] simulators, respectively. The network interface (NI) parameters, including area, power, and latency, are taken from [44]. Additionally, the area and power consumption of the D2D (Die-to-Die) interface are based on [56]. The wire model in interposer is taken from [40]. Collectively, these sources establish a consistent experimental foundation and ensure accurate modeling.

TABLE II. PARAMETERS OF THE EXPERIMENTAL PLATFORMS

GPU chiplets	
Processor cores/clusters	15
Warp size	32
Shared memory/processor cores	48KB
Texure cache size	8KB
Constant cache size	12KB
L1 data cache	16KB
L1 instruction cache	2KB
Memory per chiplet	
Bandwidth/memory module	8 Bytes / cycle
DRAM request queue capacity	32
Memory controller	Out of order (FR-FCFS)
x86 chiplets	
Core frequency	3GHz
Main memory size	2GB
Get/Decode/Submit Size	4/4/4
ROB size	64
L1 D cache (private)	16 KB
L1 I cache (private)	32 KB
L 2 cache (shared)	64 KB
Inter- and intra-chiplet network	
Flit size	256 bits
Packet size	5
Network size	4×9, 4×16, and 4×36
Number of virtual channels	1/2/4
Input buffer size	2/4/8
Routing algorithm	XY-XY (Mesh-Mesh), XY-Torus XY(Mesh-Torus), routing table(other topologies)
Random traffic pattern	uniform, shuffle, bit traversal
Packet injection rate	0.02-0.12 packet/cycle/router
Intra-chiplet latency	router: 2 cycles, link: 1 cycle
Active interposer latency	Interposer link latency model [40], PHY latency model [41], router: 2 cycles
Passive interposer latency	Interposer link latency model [40], PHY latency model [41]
Benchmarks	
Neural networks (for GPU)	ResNet, GCN, Transformer, and CNN
Data sets	CIFAR-10, MNIST, VOC
PARSEC and SPLASH-2 (for x86)	vips, barnes, blackscholes, canneal, dedup, ferret, raytrace, fluidanimate, streamcluster, freqmine

The proposed method is evaluated against previously proposed inter- and intra-chiplet interconnection networks, including Kite [6] and ButterDonut [5], where the number of chiplets is 4 and the number of cores per chiplet is 16. Additionally, the proposed method is compared against:

1) On-chip network topology generation which uses genetic algorithm (GA) and Tabu. These algorithms are modified to consider a uniform partitioning of tiles into *m* chiplets, with specific two D2D interface configurations.

2) Inter- and intra-chiplet interconnection networks based on well-known NoC topologies, which are categorized as Mesh-Mesh (both the inter- and intra-chiplet topologies are mesh), and Mesh-Torus (the inter-chiplet topology is mesh and the intra-chiplet topology is torus), and Mesh-Butterfly (the inter-chiplet topology is mesh and the intra-chiplet topology is generated by [48]).

VI.B EXPERIMENTAL RESULTS

A. Measuring the Error of the Proposed Robustness Model

The γ model defined in Eqn. (1) has been evaluated for accuracy. The robustness values computed by Eqn. (1) in the model γ_p is compared against that obtained through simulation γ_a. The regression error is defined as follows:

$$e = \left|\frac{\gamma_p - \gamma_a}{\gamma_a}\right| \times 100\%$$

Fig. 11 shows the error of the γ model. One can see that the average error is 8.15%, which indicates that the accuracy of the γ model exhibits a fairly high degree of accuracy in predicting robustness values. The proposed γ model is also compared against other regression models, including the models based on power, exponential, and linear functions. As shown in Table III, the proposed γ model yields the lowest average among all tested models, while the others have an error rate exceeding 10%.

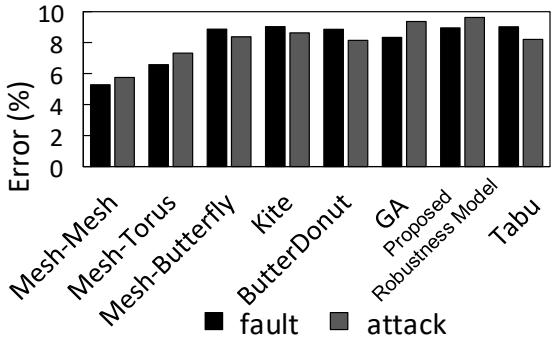


Fig. 11. Error of the proposed robustness model in Eqn. (1).

TABLE III. THE ERRORS OF DIFFERENT REGRESSION MODELS

Model	Form	Error
Power	$f(x) = a\Upsilon^b$	25.7%
Proposed	$f(x) = a\Upsilon + b$	5.3%
Exponential	$f(x) = a(\exp(b\Upsilon))$	12.8%
Linear fitting	$f(x)=a(\sin(\Upsilon\text{-}pi))+b(\Upsilon\text{-}10)^2+c$	11.3%

B. Performance Evaluation with Random Benchmarks

The proposed method has been applied to the random traffics with packet injection rates ranging from 0.02-0.2 packet per cycle per router, and the results are shown in Fig. 12.

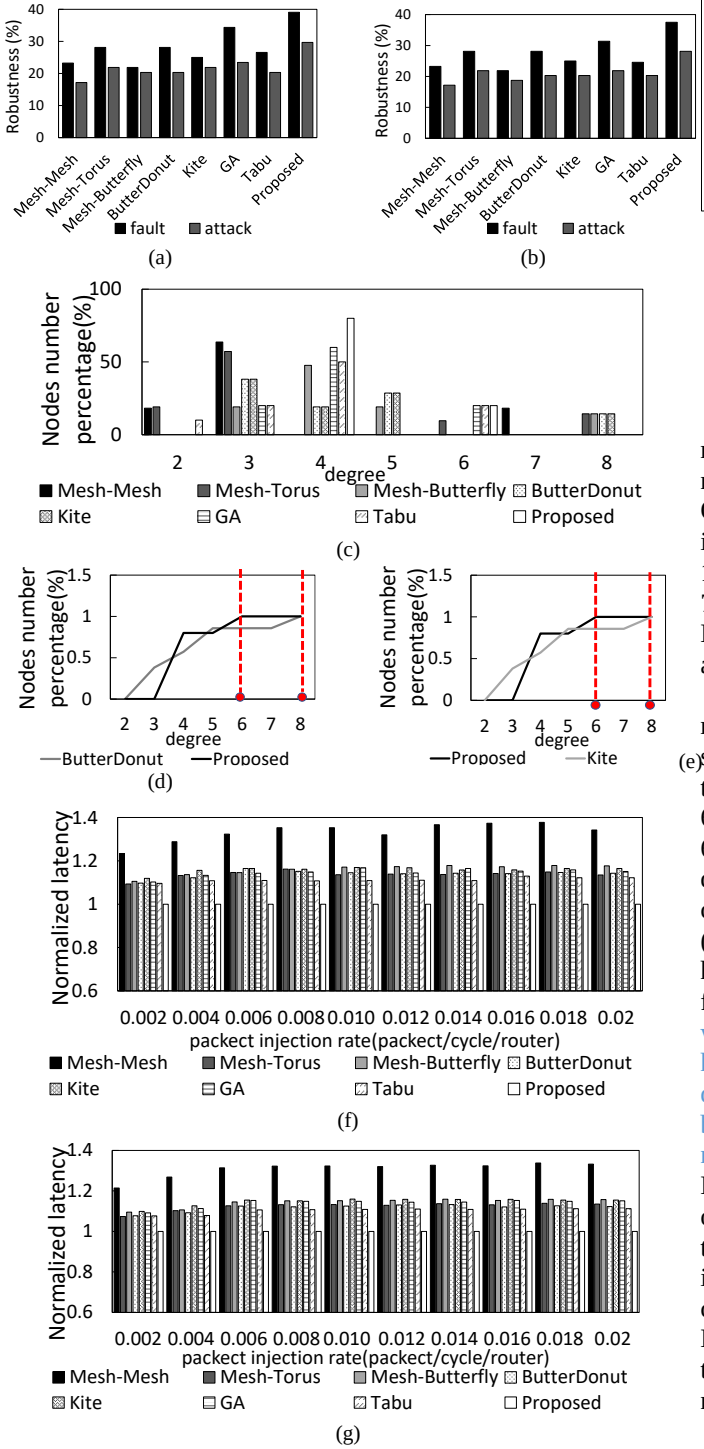


Fig. 12. (a) Robustness with active interposer. (b) Robustness with passive interposer and random traffics, where the packect injection rate is 0.002-0.02 packet/cycle/router. (c) The node distributions of different topologies. (d) The cumulative distributions of ButterDonut and the proposed method. (e) The cumulative distributions of Kite and the proposed method. (f) Normalized communication latencies with active interposer and (g) passive interposer.

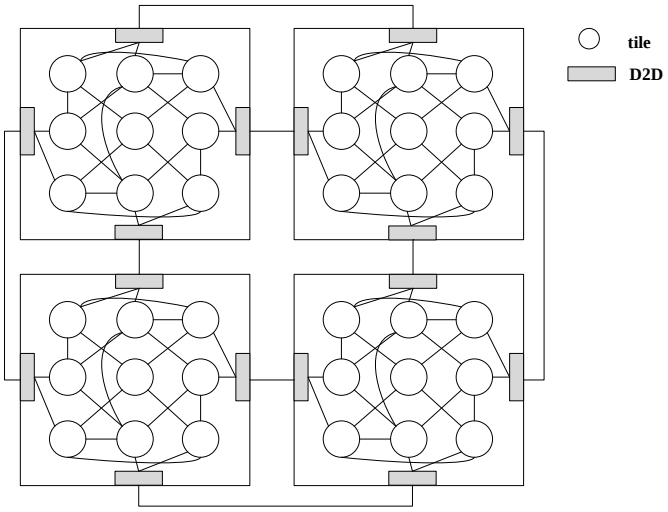


Fig. 13. The generated topology with a total tile count of 36.

Here, we examine the case with a total of 64 tiles. The robustness, area, power and communication latency of other networks are normalized to those of our proposed method. One can see from Fig. 12 (a) that the proposed method increases the robustness by 15.76%, 10.88%, 17.13%, 10.88%, 14.06%, 4.65%, 12.5% and 12.5%, 7.81%, 9.37%, 9.37%, 7.81%, 6.24%, 9.37% over Mesh-Mesh, Mesh-Torus, Mesh-Butterfly, ButterDonut, Kite, GA, Tabu under random faults and targeted attacks with active interposer, respectively.

Similar improvements in robustness are observed under random faults and target attacks with a passive interposer, as shown Fig. 12 (b). Specifically, the proposed method increases the robustness by 14.26%, 9.38%, 15.62%, 9.38%, 12.5%, 6.13%, 12.94% and 10.95%, 6.25%, 9.38%, 7.82%, 7.82%, 6.25%, 7.82%. The proposed topology's more uniform distribution of node degrees is identified as a key factor contributing to its enhanced robustness, as shown in Fig. 12 (c). In particular, 80% of nodes have a degree of 4, while 20% have a degree of 6, which allows the network to withstand failures more effectively. Note that having too many nodes with high degree raises area and power consumption, while having too many nodes with low degree reduces robustness. A degree of 4 is thus a balanced choice, offering a good trade-off between robustness and area. Consequently, most nodes in the network converge to a degree of 4. Although the Kite and ButterDonut topologies can reach a maximum degree of 8, only 14.2% of their nodes do so—fewer than in the proposed topology, where 20% of nodes reach a degree of 6. As shown in Figures 12(d) and 12(e), the proposed topology reaches a cumulative distribution probability of 1 at degree 6, whereas Kite and ButterDonut both do so at degree 8. Once nodes with the highest degree become targets of an attack, overall network performance declines significantly.

From Fig. 12 (f) and (g), one can see that with the increase of injection rate, the network latency increases gradually. However, the proposed method always has the minimal latency, *i.e.*, it reduces the communication latency by 30.8%, 12.4%, 14.3%, 11.7%, 14.75%, 13.95%, 10.3% and 33.3%, 13.7%, 16%, 13.9%, 15.9%, 14.6%, 11.2% over Mesh-Mesh, Mesh-Torus, Mesh-Butterfly, ButterDonut, Kite, GA, Tabu with active and passive interposers, respectively.

The generated topology with a total tile count of 36 is shown in Fig. 13. This topology can be implemented using existing chiplet integration technologies.

C. Sensitivity Analysis

The proposed method consistently improves network robustness when compared to ButterDonut, Kite, GA, and Tabu under various total tile counts (36, 64, and 144) under random faults and targeted attacks, as depicted in Fig. 14. In Fig. 14 (a), under random faults, the proposed method increases the robustness by 8.33%, 11.1%, 5.55%, 8.33% over ButterDonut, Kite, GA, Tabu with a total tile count of 36. It also increases the robustness by 10.94%, 14.06%, 4.69%, 12.5% over ButterDonut, Kite, GA, Tabu with a total tile count of 64, and 9.89%, 8.90%, 7.68%, 10.98% with a total tile count of 144. Moving on to Fig. 14 (b), under targeted attacks, one can see that the proposed method increases robustness by 7%, 5.2%, 5.6%, 8.2% and 9.37%, 7.81%, 6.24%, 9.37% and 7.13%, 8.69%, 8.69%, 7.15% over ButterDonut, Kite, GA, Tabu with total tile counts of 36, 64, and 144, respectively. It is worth noting that the network robustness under targeted attacks is significantly lower than that under random faults.

Targeted attacks specially disable nodes with the highest degree, leading to a fast network disconnection.

As shown in Fig. 14 (c), the proposed method reduces latency by 2.41%, 6.29%, 4.52%, 6% and 17.25%, 10.88%, 6.48%, 8% and 15.78%, 11.06%, 8.99%, 7% over ButterDonut, Kite, GA, Tabu with total tile counts of 36, 64, and 144, respectively. This indicates that the network performance improvement due to the proposed method becomes more pronounced with larger node sizes.

Evaluation results regarding the impact of VC (virtual channel) number and buffer size on network latency are shown in Fig. 14 (d) and (e). In this assessment, we consider cases with VC numbers of 4, 6, and 8 and buffer sizes of 2, 4, 8. The proposed method decreases latency by 13.76%, 18.14%, 9.86%, and 16% over ButterDonut, Kite, GA, Tabu with VC numbers of 4, 6 and 8. It also decreases latency by 14.97%, 14.61 %, 14.61% and 19.66%, 18.99%, 18.99% over ButterDonut and Kite with buffer sizes of 2, 4 and 8, respectively. The results suggest that the number of virtual channels and buffer size have minimal impact on network latency.

Finally, Fig. 14(f) presents experimental results on how the number of D2D interfaces affects network robustness. Three configurations are evaluated, with D2D counts of 1, 2, and 4. As shown in Fig. 14(f), increasing the number of D2D interfaces bolsters network robustness. For a network with 36 tiles, the proposed method's robustness exceeds the configurations with 1 and 2 D2D interfaces by 16.12% and 9.3%, respectively. In a 64-tile network, those increases stand at 18.15% and 9.66%, respectively, and in a 144-tile network, they reach 23.15% and 12.25%. Notably, as the network size grows, the robustness benefits become more substantial. This indicates that additional D2D interfaces provide more alternative paths and enhanced connectivity, both of which are essential for preserving performance and fault tolerance in larger topologies.

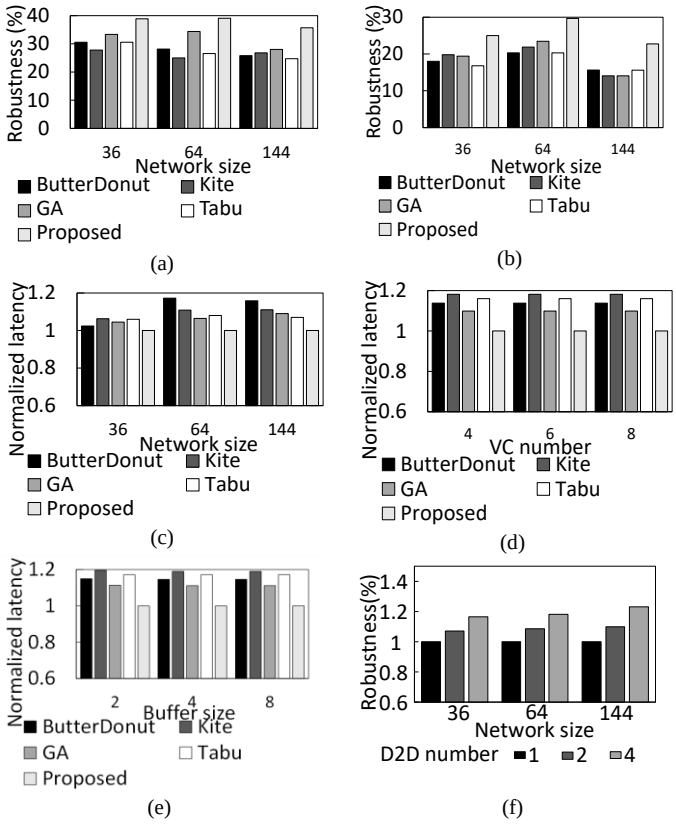


Fig. 14. (a) Robustness under random faults; (b) robustness under targeted attacks; (c) communication latencies with different node sizes; (d) communication latencies with different VC numbers; (e) communication latencies with different buffer sizes; (f) robustness with different D2D number and network sizes.

D. Performance Evaluation Using PARSEC and SPLASH2 Benchmarks

Performance of the proposed method using PARSEC and SPLASH2 benchmarks are assessed, and the results are shown in Fig. 15. In this evaluation, we exclusively focus on the execution and communication latency of the network, since the robustness, area, and power consumption of each network are primarily influenced by the topology structures, rather than the benchmarks themselves. Fig. 15 (a) and (b) demonstrate that the proposed method consistently reduces the execution time, achieving improvements of 12.4%, 11.9%, 10.81%, 10.46%, 9.8%, 9.5%, 9.79% and 11.51%, 10.67%, 9.61%, 8.96%, 8.52%, 8.05%, 8.99% over Mesh-Mesh, Mesh-Torus, Mesh-Butterfly, ButterDonut, Kite, GA, Tabu, considering both active and passive interposers, respectively. Moving on to Fig. 15 (c) and (d), one can see that the proposed method also reduces communication latency effectively, achieving reductions of by 25.3%, 11.1%, 9.86%, 8.34%, 13%, 9.9%, 9% and 23.9%, 10.7%, 9.35%, 7.54%, 12.75%, 9.5%, 8.5% over Mesh-Mesh, Mesh-Torus, Mesh- Butterfly, ButterDonut, Kite, GA, Tabu, considering both active and passive interposers, respectively. These results underscore the efficacy of our approach in enhancing network performance across a range of benchmarks and network configurations.

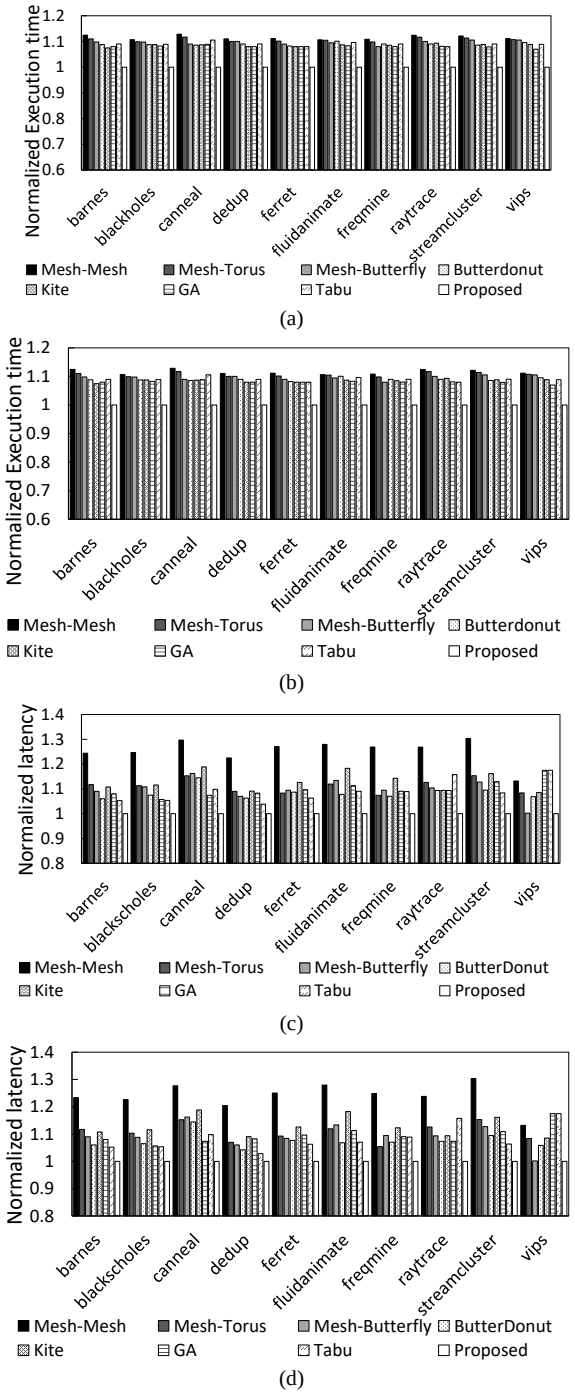


Fig. 15. (a) Application execution times with active interposer and (b) passive interposer. (c) Communication latencies with active interposer and (d) passive interposer running PARSEC and SPLASH2 benchmarks.

E. Performance Evaluation with Neural Network Applications

We evaluate different networks with different neural network benchmarks in Fig. 16. In Fig. 16 (a) and (b), it is evident that the proposed method reduces the execution time, achieving improvements of 21.59%, 15.7%, 12.65%, 8.37%, 6.94%, 8.21%, 8.43% and 19.59%, 12.95%, 9.65%, 7.62%, 4.94%, 7.91%, 8.41% over Mesh-Mesh, Mesh-Torus, Mesh-Butterfly, ButterDonut, Kite, GA, Tabu with active and passive interposers, respectively. Moving on to Fig. 16 (c) and (d), one can see that the proposed method also leads to a notable reduction in communication latency, with improvements of 23.79%, 17.5%, 14.64%, 10.87%, 9.19%, 9.66%, 9.53 % and 21.84%, 16.9%, 13.64%, 9.62%, 7.94%, 8.91%, 9.16% over Mesh-Mesh, Mesh-Torus, Mesh-Butterfly, ButterDonut, Kite, GA, Tabu, considering both active and passive interposers, respectively.

F. Performance Evaluation on FPGA-Based Multicore System

We evaluated the proposed method on a multicore system on Xilinx vu3p FPGA. The multicore system uses PULPino [42] as the processor cores, and OpenPiton [48] as the uncore (NoC and NoI, memory, *etc.*). The configurations of PULPino and OpenPiton follow the default configurations from [49] and

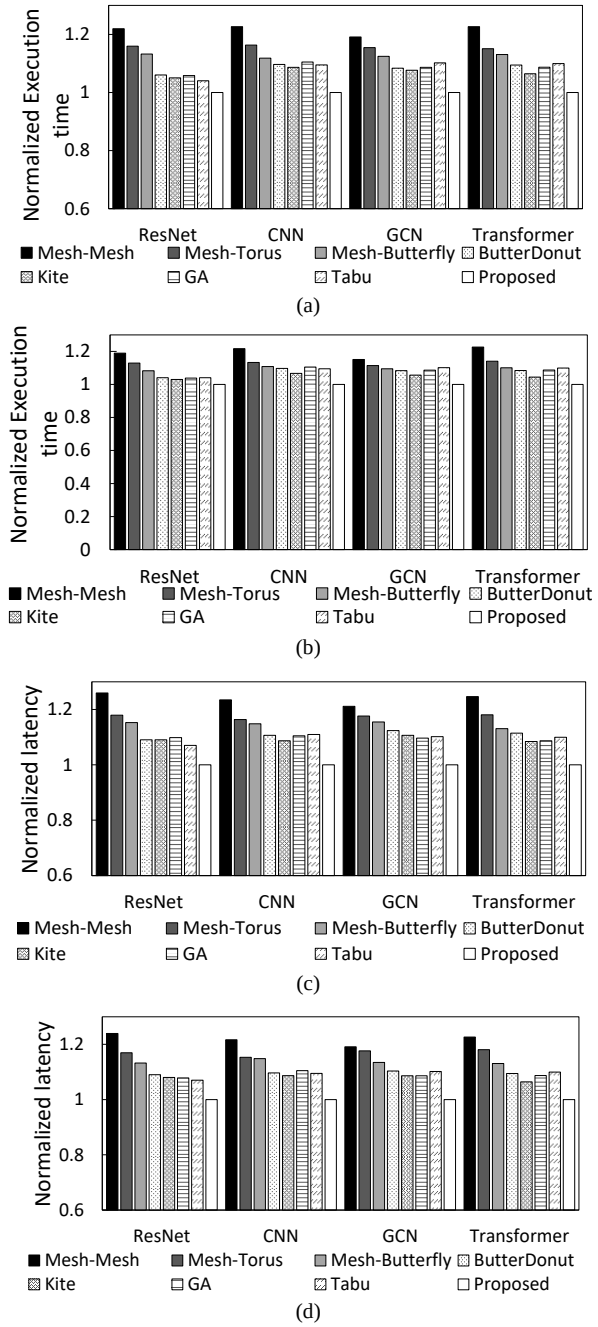


Fig. 16. (a) Application execution times with active interposer and (b) passive interposer. (c) Communication latencies with active interposer and (d) passive interposer running different neural network applications.

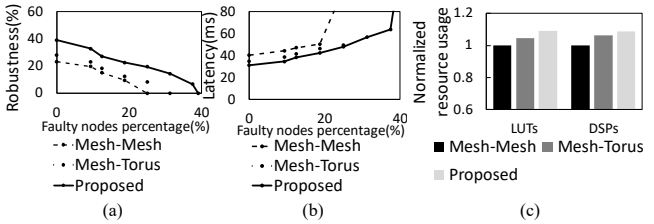


Fig. 17. (a) Robustness and (b) communication latencies with different percentages of faulty nodes. (c) Resources utilization of the networks.

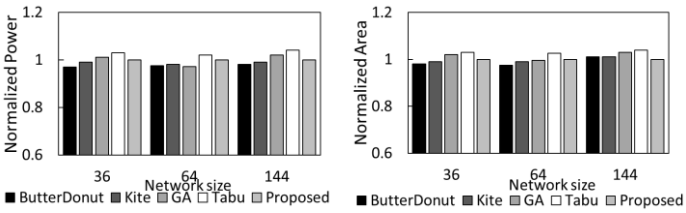


Fig. 18. (a) Normalized areas and (b) powers with different network sizes.

[50], respectively. The configuration of the inter- and intra-chiplet network is the same as in Table II. The robustness and latency of the proposed method with the different proportions of faulty nodes are shown in Fig. 17. From Fig. 17 (a) and (b), one can see that the proposed method increases the robustness by 15.82%, 10.8% and reduces the communication latency by 29.3%, 11.5% over Mesh-Mesh, Mesh-Torus, respectively. From Fig. 17 (c), one can see that resource utilization of the proposed method is slightly higher than Mesh-Mesh, Mesh-Torus, however, the resource usage is still within the threshold (the maximum resource of the FPGA).

G. Area and Power Evaluation

Area and power of the proposed method are compared and shown in Fig. 18. Fig. 18 (a) and (b) demonstrate that the proposed method has similar area and power consumption compared to other topologies.

VII. Conclusion

Inter- and intra-chiplet interconnection networks serve as the backbone of multi-chiplet systems, providing the essential framework to enable efficient communication, scalability, fault tolerance, and performance optimization. In this paper, robustness of inter- and intra-chiplet interconnection networks was modelled and characterized by various network parameters, including the average clustering coefficient (*ACC*), average shortest path length (*APL*), average neighbor degree (*AND*), the most frequent degree value of topology (k_{most}), and the highest degree value of topology (k_{max}) . An optimization problem was subsequently formulated, aimed at generating network topologies that maximize robustness while adhering to the power and area constraints. To tackle this problem, an efficient reinforcement learning algorithm was proposed. Experimental results demonstrated that our proposed method significantly enhances network robustness. Under random faults, our approach achieved improvements of 15.76% over Mesh-Mesh, 10.88% over Mesh-Torus, 17.13% over Mesh-Butterfly, 10.88% over ButterDonut, 14.06% in Kite, 4.65% over GA, and 12.5% over Tabu.. These outcomes underscore the suitability of our approach for generating resilient large scale inter- and intra-chiplet networks, which are essential for the development and deployment of multi-chiplet based many-core systems.

References

[1] S. Shamshiri, A. Ghofrani and K.-T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in ITC, 2011, pp. 1-10.

[2] N. Vashistha, M. M. Al Hasan, N. Asadizanjani, F. Rahman and M. Tehranipoor, “Trust validation of chiplets using a physical inspection-based certification authority,” in ECTC, 2022, pp. 2311-2320.

[3] V. Y. Raparti and S. Pasricha, “Lightweight mitigation of hardware Trojan attacks in NoC-based manycore computing,” in DAC, 2019, pp. 1-6.

[4] J. Harttung, E. Franz, S. Moriam, and P. Walther, “Lightweight authenticated encryption for network-on-chip communications,” in GLVLSI, 2019, pp. 33-38.

[5] A. Kannan, N. D. Enright Jerger, G. H. Loh, “Enabling interposer-based disintegration of multi-core processors,” in MICRO, 2015, pp. 546-558.

[6] S. Bharadwaj, J. Yin, B. M. Beckmann, T. Krishna, “Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling,” in DAC, 2020, pp. 1-6.

[7] W. J. Dally, B. Towles, “Route packets, not wires: on-chip interconnection networks,” in DAC, 2001, pp. 684-689.

[8] W. J. Dally, C. L. Seitz, “The torus routing chip,” in Distributed Computing, 1986, pp. 187-196.

[9] N. Enright Jerger, A. Kannan, Z. Li, and G. H. Loh, “NoC architectures for silicon interposer systems,” in MICRO, 2014, pp. 458-470.

[10] K. S. -M. Li, S. -Y. Chen, L. -B. Chen and R. -T. Gu, “A fast custom network topology generation with floorplanning for NoC-based systems,” in IEEE IC Design & Tech. Conf., 2011, pp. 1-4.

[11] K. Srinivasan, K. S. Chatha, G. Konjevod, “Linear-programming-based techniques for synthesis of network-on-chip architectures,” in VLSI Systems, 2006, pp. 407-420.

[12] N. Venkataraman and R. Kumar, “Design and analysis of application specific network on chip for reliable custom topology,” in Computer Networks, vol. 158, pp. 69-76, 2019.

[13] C. Neeb and N. When, “Designing efficient irregular networks for heterogeneous systems-on-chip,” in J. Syst. Archit., vol. 54, no. 3-4, 2008, pp. 384-396.

[14] P. Yang, Q. Wang, W. Li, Z. Yu and H. Ye, “A fault tolerance NoC topology and adaptive routing algorithm,” in ICESS, 2016, pp. 42-47.

[15] M. Hosseinabady, M. R. Kakoee, J. Mathew and D. K. Pradhan, “Reliable network-on-chip based on generalized de Bruijn graph,” in HLDVT, 2007, pp. 3-10.

[16] M. Wang, Y. Wang, C. Liu and L. Zhang, “Network-on-interposer design for agile neural-network processor chip customization,” in DAC, 2021, pp. 49-54.

[17] F. Li, Y. Wang, Y. Cheng, Y. Wang, Y. Han, H. Li, and X. Li, "GIA: A reusable general interposer architecture for agile chiplet integration.” in ICCAD, 2022, pp. 1–9.

[18] H. Sharma, S. K. Mandal, J. R. Doppa, U. Y. Ogras and P. P. Pande, "SWAP: A server-scale communication-aware chiplet-based manycore PIM accelerator," in IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 41, no. 11, 2022, pp. 4145-4156.

[19] H. Sharma, L. Pfromm, R. O. Topaloglu, J. R. Doppa, U. Y. O. A. Kalyanraman, and P. P. Pande, “Florets for chiplets: Data flow-aware high-performance and energy-efficient network-on-interposer for CNN inference tasks,” in ACM Trans. Embed. Comput. Syst. vol.22, no. 132, 2023.

[20] J. Yin, et al., “Modular routing design for chiplet-based systems,” in ISCA, 2018, pp. 726-738.

[21] E. Taheri, S. Pasricha and M. Nikdast, “DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5d chiplet networks,” in DATE, 2022, pp. 1047-1052.

[22] P. Majumder, S. Kim, J. Huang, K. H. Yum and E. J. Kim, “Remote control: A simple deadlock avoidance scheme for modular systems-on-chip,” in IEEE Tran. Computers, vol. 70, no. 11, 2021, pp. 1928-1941.

[23] M. Sinha, S. Gupta, S. S. Rout, and S. De., “Sniffer: A machine learning approach for DoS attack localization in NoC-based SoCs,” in IEEE J. Emerg. Sel. Topics Circuits Syst., vol. 11, no. 2, 2021, pp. 278-291.

[24] L. Zhang, X. Wang, Y. Jiang, M. Yang, T. Mak, and A. K. Singh, “Effectiveness of HT-assisted sinkhole and blackhole denial of service attacks targeting mesh networks-on-chip,” in J. Syst. Archit., vol. 89, 2018, pp. 84-94.

[25] V. Janfaza and E. Baharlouei, “A new fault-tolerant deadlock-free fully adaptive routing in NOC,” in EWDTS, 2017, pp. 1-6.

[26] A.-László Barabási, “Network science,” Cambridge University Press. 2016.

[27] J. Friedman, T. Hastie, R. Tibshirani, “The elements of statistical learning,” Springer, 2001.

[28] B. S. Manoj, A. Chakraborty, R. Singh, “Complex networks: a networking and signal processing perspective,” O’Reilly, 2019.

[29] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770- 778.

[30] A. Vaswani, N. Shazeer, N. Parmar, et al., “Attention is all you need,” in NIPS, 2017.

[31] T. N. Kipf, M. Welling, “Semi-supervised classification with graph convolutional networks,” in ICLR, 2017.

[32] Simple CNN, Available: https://github.com/can1357/simple_cnn, 2017.

[33] CIFAR-10, Available: https://tensorflow.google.cn/datasets/catalog/cifar10.

[34] MNIST, Available: http://yann.lecun.com/exdb/mnist/.

[35] VOC, Available: http://host.robots.ox.ac.uk/pascal/VOC/.

[36] M. Wang, et al. “Network-on-interposer design for agile neural- network processor chip customization,” in DAC, 2021.

[37] A. A. Huqqani, et al. “Multicore and GPU parallelization of neural networks for face recognition,” in ICCS, 2013.

[38] H. Zhi, X. Xu, W. Han, Z. Gao, et al, “A methodology for simulating multi-chiplet systems using open-source simulators,” in NANOCOM, 2021, pp. 1-6.

[39] M. Khairy, et al. “Accel-aim: An extensible simulation framework for validated GPU modeling,” in ISCA, 2020.

[40] M. A. Kabir and Y. Peng, “Chiplet-package co-design for 2.5D systems using standard ASIC cad tools,” in ASP-DAC, 2020, pp. 351-356.

[41] D. D. Sharma, et al. “Universal chiplet interconnect express (UCIe)™: building an open chiplet ecosystem,” in IEEE Trans. Compon. Packag. Manuf. Technol., vol. 12, no. 9, 2022, pp. 1423-1431.

[42] A. Traber, F. Zaruba, S. Stucki, et al., “PULPino: a small single-core RISC-V SoC,” 3rd RISCV Workshop.

[43] Volta V100 White Paper, Available: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

[44] M. Schoeberl M, Pezzarossa L, Spars J, “A minimal network interface for a simple network-on-chip,” in ARCS, 2019, pp. 295-307.

[45] P. Vivet et al. “2.3 A 220GOPS 96-core processor with 6 chiplets 3D-stacked on an active interposer offering 0.6ns/mm latency, 3Tb/s/mm^2 inter-chiplet interconnects and 156mW/mm^2@ 82%-peak-efficiency DC-DC converters,” in ISSCC, 2020.

[46] C. Sun et al., “DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling,” in NoCS, 2012, pp. 201-210.

[47] N. Muralimanohar, R. Balasubramonian, N. P. Jouppi, “CACTI 6.0: a tool to model large caches,” HP laboratories, 2009.

[48] J. Balkind, M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, A. Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff, “OpenPiton: an open source manycore research framework.” in SIGPLAN, 2016, pp. 217-232.

[49] Pulpino, Available: https://www.pulp-platform.org/.

[50] Openpiton, Available: http://parallel.princeton.edu/openpiton.

[51] A. Saffidine, T. Cazenave and J. Méhat, "UCD: upper confidence bound for rooted directed acyclic graphs," in Knowl. Based Syst., 2010, pp. 467-473.

[52] D. P. Kingma, J. Ba, "Adam: a method for stochastic optimization," in Proceedings of ICLR'15, 2015.

[53] G. Li, et al., "DeepGCNs: can GCNs go as deep as CNNs?" in CVPR, 2020, pp. 5567-5576.

[54] X. Glorot, A. Bordes, and Y. Bengio. "Deep sparse rectifier neural networks," in AISTATS, 2011.

[55] M. Leshno, et al., "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," in Neural networks, vol. 6, no. 6, 1993, pp. 861-867.

[56] Y. Feng, D. Xiang, K. Ma, “Heterogeneous die-to-die interfaces: Enabling more flexible chiplet interconnection systems”, in MICRO, 2023, pp. 930-943.

[57] B. Ye, et al., “A 2.29-pJ/b 112-Gb/s wireline transceiver with RX four-tap FFE for medium-reach applications in 28-nm CMOS,” in IEEE J. Solid-state Circuits, vol. 58, no. 1, 2023, pp. 19–29.

[58] Y. Hu, X. Lin, et al., “Wafer-scale computing: advancements, challenges, and future perspectives,” in IEEE Circuits Syst. Mag, vol. 24, 2024, pp. 52-81.

[59] Y. Han, et al., "The big chip: challenge, model and architecture," in Fundamental Research, vol. 4, pp. 1431-1441, 2024.

[60] S. Li, J. H. Ahn, R. D. Strong, et al., "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in MICRO, 2009, pp. 469-480.

[61] F. Schätzle, et al., “Modeling methodology for multi-die chip design based on gem5/systemC co-simulation,” in RAPIDO@HiPEAC, 2024, pp. 35-41.

[62] X. Li, et al., “MUG5: Modeling of universal vhiplet interconnect express (UCIe) standard based on gem5,” in ASICON, 2023, pp. 1-4.

Xiaohang Wang Xiaohang Wang received the B. Eng. and Ph. D. degree in communication and electronic engineering from Zhejiang University, in 2006 and 2011, respectively. He is currently a professor at Zhejiang University. He was the receipt of PDP 2015 and VLSISoC 2014 Best Paper Awards. His research interests include many-core architecture, power efficient architectures, optimal control, and NoC-based systems.

Miao Xu Miao Xu received his bachelor degree in software engineering from Yunnan University (YNU), Kunming, China. She is pursuing her master degree in the school of software engineering, SCUT. Her research interests include multi-chiplet and design space exploration.

Amit Kumar Singh Amit Kumar Singh (M’09) is a Lecturer at University of Essex, UK. He received the B.Tech. degree in Electronics Engineering from Indian Institute of Technology (Indian School of Mines), Dhanbad, India, in 2006, and the Ph. D. degree from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore, in 2013. He was with HCL Technologies, India for year and half until 2008. He has a postdoctoral research experience for over five years at several reputed universities. His current research interests are system level design-time and runtime optimizations of 2D and 3D multi-core systems for performance, energy, temperature, reliability and security. He has published over 80 papers in reputed journals/conferences, and received several best paper awards, e.g. ICCES 2017, ISORC 2016 and PDP 2015. He has served on the TPC of prestigious IEEE/ACM conferences DAC, DATE, CASES and CODES+ISSS.

Yingtao Jiang Yingtao Jiang received his Ph. D. in Computer Science from the University of Texas at Dallas in 2001. Upon graduation, he immediately joined the Department of Electrical and Computer Engineering (ECE), University of Nevada, Las Vegas, where he was promoted to full professor in 2013, and subsequently served as the ECE Department Chair between 2015 and 2018. Currently, he is the associate dean of the college of engineering at the same university. His research interests include algorithms, computer architectures, VLSI, networking, nanotechnologies, etc.

Mei Yang Mei Yang received her Ph. D. in Computer Science from the University of Texas at Dallas in Aug. 2003. In Aug. 2004, she joined in the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, where she was promoted to full professor in 2016. Her research interests include computer architectures, interconnection networks, machine learning, and embedded systems. Letian Huang Letian Huang received the MS and Ph. D. degrees in communication and information system from the University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2009 and 2016, respectively. He is an associate professor with UESTC. His scientific work contains more than 40 publications including book chapters, journal articles and conference papers. His research interests include heterogeneous multi-core system-on-chips, network-on-chips, and mixed signal IC design.