2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-7652-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/HPCA56546.2023.10070981

A Scalable Methodology for Designing Efficient Interconnection Network of Chiplets

Yinxiao Feng
Institute for Interdisciplinary Information Sciences (IIIS)
Tsinghua University
Beijing, China
fyx20@mails.tsinghua.edu.cn

Dong Xiang*
School of Software
Tsinghua University
Beijing, China
dxiang@tsinghua.edu.cn

Kaisheng Ma*
Institute for Interdisciplinary Information Sciences (IIIS)
Tsinghua University
Beijing, China
kaisheng@mail.tsinghua.edu.cn

***Abstract*—The *Chiplet* methodology can accelerate VLSI system development and provide better flexibility. However, it is not easy to build interconnection networks across multiple chiplets and maintain high-performance deadlock-free routing in systems of various hierarchical topologies. In particular, most on-chiplet networks are based on flat topologies such as 2D-mesh, which are inflexible and insufficient for large-scale multi-chiplet systems.**

To take full advantage of the multi-chiplet architecture and advanced packaging, we propose an interconnection method that can flexibly establish high-radix interconnection networks from typical 2D-mesh-NoC-based chiplets. A minus-first-based deadlock-free adaptive routing algorithm and a *safe/unsafe* flow control policy are introduced for these multi-chiplet interconnection networks. Additionally, a general approach *network interleaving* is used to balance the communication bandwidth within and between chiplets.

We evaluate different architectures and traffic patterns on a cycle-accurate C++ simulator. Compared with traditional adaptive routing in 2D-mesh, our methodology can significantly improve network performance in various cases. The more chiplets there are, the more effective the method is. For 64 4×4-2D-mesh-based chiplets, The maximum injection rate increase is up to 2×, and the average latency reduction is up to 45%.

***Index Terms*—Chiplet, network-on-chip, interconnection network, adaptive routing, deadlock-free.**

I. Introduction

Multi-chiplet architecture has become one of the most popular VLSI design methodologies in recent years. As shown in Fig. 1, with the increasing progress of advanced packaging and high-speed wireline technologies, multiple chiplets can be interconnected at very high wiring density and communication bandwidth. Many fantastic works from academia and industry have adopted the multi-chiplet architecture that is believed to provide higher performance and scalability [1]–[12]. However, the interconnection network designs of these systems remain primitive and benefit little from the chiplet architecture. They rarely use long connection links or topologies with smaller diameters. At the same time, many network designs can only be adapted to a specific system. Building chiplet-based interconnection networks still face many challenges.

One challenge is building systems at different scales with the same chiplet. Chiplet reuse among different systems is one of the attractive features of chiplet architecture [17]. If a chiplet can be used to build multiple systems, significant design cost and iteration time will be saved [18]. 2D-mesh topology is widely used in on-chip networks because it is implement-friendly [19]–[23]. However, for large systems with thousands of cores, such as the wafer-level system [7]–[12], the performance of 2D-mesh networks is poor because the diameter is up to $O(\sqrt{N})$. There are significant differences in design approaches between large and small networks; thus, it is challenging to develop systems of various scales based on the same chiplet.

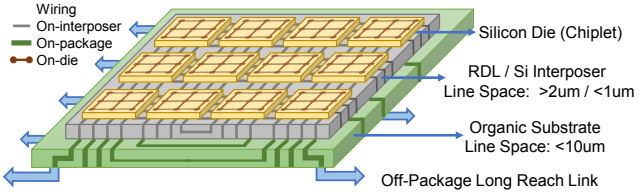

Fig. 1. Multi-chiplet system. Besides the on-die wiring, the on-RDL, on-silicon, on-substrate, and off-package wiring resources can support more complex interconnections [13]–[16].

Another challenge is the routing design for multi-chiplet systems. The interconnection network is particularly sensitive to routing issues. However, conventional routing technologies cannot be directly applied to multi-chiplet systems. Connecting several networks-on-chip (NoCs) together can lead to potential deadlocks and congestion problems [24] [25]. At the same time, the routing algorithm is closely related to the topology and scale of the system, so it is not easy to expand. Almost all recent works adopt simple but inefficient interconnection and routing schemes such as dimensional-order routing (DOR) on 2D-mesh [4] [6]–[12] [26]–[28]. Such primitive approaches not only limit performance and scalability but also fail to provide fault tolerance.

A few works have made efforts on these issues. For the routing challenge, Yin *et al.* presented a new routing method for multi-chiplet systems based on turn restrictions [24], but this approach requires significant modifications to hardware and can lead to imbalance traffic. Majumder *et al.* proposed *Remote Control* to solve deadlock problems [25], but large

*Corresponding authors

buffers are needed to store all outbound packets. There are also deadlock recovery techniques such as *Pitstop* [29] and *UPP* [30], but they all need custom hardware and spare resources. These works provide solutions to cross-chiplet deadlocks but do not provide a high-performance and scalable solution for multi-chiplet interconnections.

As for the scalability challenge, Zheng *et al.* proposed *Adapt-NoC* with adaptable links and routers to change the topologies of the sub-networks, but such adaptation is very limited and the topologies are still flat [31]. Shao *et al.*presented a scalable deep neural network (DNN) accelerator called Simba based on hierarchical 2D-mesh [4] [6]. Other scalable chiplet-based systems in industry are also based on 2D-mesh [7] [8] [28]. Although these works have achieved sizeable systems by a single chiplet, they have not fully exploited advanced packaging technologies to design more efficient interconnection networks. As far as we know, there is no work to provide a high-performance and scalable chiplet interconnection solution.

To address the challenges, we propose a new software-based approach that can redefine the topological property of the traditional 2D-mesh-NoC-based chiplet. The interface nodes of chiplets are grouped into several abstract interfaces, and high-radix topologies can be flexibly interconnected. For such 2D-mesh-NoC-based high-radix networks, a scalable deadlock-free adaptive routing algorithm is presented based on minus-first routing (MFR). Additionally, we propose the *safe/unsafe* flow control to simplify routing design and the *network interleaving* method to improve the bandwidth usage of the interface. The contributions of this paper can be summarized as follows:

- We present a method to scale large high-radix interconnection systems with 2D-mesh-NoC-based chiplets, achieving more efficient and flexible interconnection without changing much of the typical NoC architecture.
- We propose a scalable deadlock-free adaptive routing algorithm for interconnection networks of chiplets. The algorithm is applicable to most common topologies, including nD-mesh, hypercube, and dragonfly. Compared with traditional MFR, the new algorithm is modularly labeled and easy to scale at chiplet level.
- We introduce the *safe/unsafe* flow control and the *network interleaving* method to address potential drawbacks arising from the new interconnection design. The two methods are general and can be applied to most multi-channel interconnection networks.
- We evaluate different architectures and traffic patterns on a cycle-accurate C++ simulator. The evaluation results show that our methodology has significant performance advantages over traditional multi-chiplet interconnection networks based on flat topologies.

II. Background

A. Multi-Chiplet Architecture

The conventional VLSI system is implemented on a monolithic die. However, the chip area is limited by the lithographic reticle, and designing large chips is very costly [1] [32]. Multi-chiplet architecture is to build large systems from multiple small chiplets. The design difficulty and manufacturing cost of small chiplets are far less than that of large chips [18].

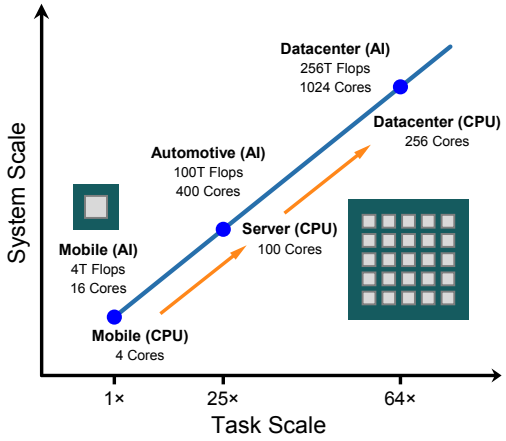

Fig. 2. Chiplet reuse in different scale systems for different workloads. In different scenarios, the same task requires different computing scales.

Moreover, packaging technology and high-speed wireline technology have made great progress in recent year. *Serdes* technologies such as *OIF-CEI-112G* can interconnect chiplets at high speed over long distances on the interposer or substrate [33] [34]. 2.5D packaging technologies such as *CoWoS* provide large interposers with fine pitch and high volumes wires for chiplets integration [15]. With the support of advanced packaging and high-speed wireline technology, there are more interconnection possibilities off-chip than on-chip.

Chiplet reuse is another attractive characteristic of multi-chiplet architecture. Feng *et al.* show that significant design costs can be saved by reusing chiplets within a single system and between multiple systems [17] [18]. As shown in Fig. 2, the scale of modern computing systems varies under different work scenarios. Edge AI computer Jetson Nano contains 128 CUDA cores [35] while datacenter system DGX integrates eight 6912-core A100 [36] [37]. If a single AI chiplet can cover all scenarios from 0.5 TFLOPs to 5 PFLOPs, huge cost savings can be achieved. Zimmer *et al.* present the multi-chip-module-based DNN inference accelerator [4] [6], which can scale from 0.32 to 128 TOPS. Ignjatović *et al.* present the wormhole AI training processor [28], which can also scale from one chip into a large 2D processing grid. However, their interconnect architectures are just simple extensions of the traditional flat topologies, which are insufficient for large-scale systems and do not leverage the high-density wiring of advanced packaging. More scalable and efficient multi-chiplet interconnection architectures are urgently needed.

B. Interconnection Network Bascis

The interconnection network is an indispensable part of all computing systems. The communication between any pair of components in the system must be through the interconnection network. All collective communication operations are also

completed via the network. Therefore, cost-effective network design and efficient routing algorithms are essential to system performance.

Topology is one of the most important factors affecting network performance. As shown in Table I, for an N node network, 2D-mesh has a long diameter of $2(\sqrt{N}-1)$ while hypercube has a short diameter of $\log_2 N$. The larger the network, the larger the diameter difference caused by topology. Cerebras presents a wafer-scale system, which interconnects hundreds of dies into 2D-mesh [11] [12]. As the network diameter is so large, they have to keep the communication as localized as possible. Therefore, it is necessary to adopt higher radix topologies for large-scale systems.

TABLE I
NETWORK DIAMETER

2D-Mesh	2D-Torus	nD-Mesh	Hypercube
$2(\sqrt{N}-1)$	$\sqrt{N}$	$n(\sqrt[n]{N}-1)$	$\log_2 N$

In addition to the interconnection topology, routing algorithms are also important. In addition to ensuring high-performance connectivity of the network, the routing algorithm should also ensure deadlock-free. Virtual channels can be used to avoid deadlocks [38]. A physical channel is split into several virtual channels in a time division multiplex way to isolate channel resource dependencies. However, large and complex systems require a huge number of virtual channels, which bring serious area and power overhead. As for small systems, these virtual channels become wasteful. Therefore, we need to minimize the use of virtual channels.

Duato *et al.* proposed a sufficient condition for designing deadlock-free adaptive routing algorithms for arbitrary interconnect networks, as shown in Lemma 1 [39]. Based on this theorem, we only need to provide a deadlock-free routing algorithm as the baseline routing scheme, and other channels are used as the adaptive channels for routing flexibility.

Lemma 1. *For a virtual cut-through (VCT) switching interconnection network I, a connected routing function R is deadlock-free if there exists a channel subset $C_1 \subset C$ such that the routing subfunction $R_1(x,y) = R(x,y) \cap C_1 \forall x, y \in N$ is connected and deadlock-free.*

Minus-first routing (MFR) is a label-based algorithm developed from Up*/Down* routing [40]–[43]. Compared with Up*/Down* routing, MFR is a more general approach. MFR assigns each node a numerical label, all links from smaller labels to larger labels are plus links, and links from larger to smaller are minus links. A legal route cannot be switched from plus links to minus links. In this way, deadlocks are avoided because channel dependency cycles cannot form. As for 2D-mesh, negative-first routing (NFR), also known as turn restriction routing [44], is a special case of MFR. It restricts turning to a negative direction $(X-, Y-)$ while the current direction is positive $(X+, Y+)$. However, in traditional MFR, the whole system is uniformly labeled, and all node labels are different. If the network topology or scale changes, nodes must be re-labeled; Otherwise, deadlock-free and connectivity are not guaranteed. Therefore, traditional MFR is hard to scale for multi-chiplet systems.

C. Interleaving

Interleaving has been widely used in modern high-performnace computer systems, especially in memory systems [45] [46]. There is always an upper limit to the bandwidth or capacity of single storage; Therefore, it is necessary to introduce the concept of multi-channel. By introducing multiple controllers and storage that can be accessed concurrently, the total throughput of the system is greatly improved. However, if the addresses of multiple controllers are arranged in sequence, due to the locality of data, only one controller is active for each access [47]. The bandwidth benefits of multi-channel are underutilized. As a result, methods to interleave addresses and distribute data across multiple controllers are proposed, such as interleaved memory and RAID technology [48].

Chiplet-based interconnect networks have similar problems. The off-chip bandwidth is often more scarce than the on-chip bandwidth, which makes the inter-chiplet communication become the bottleneck of the entire interconnection network. In order to improve communication efficiency or to increase fault tolerance, the connection between two chiplets is usually more than one channel. Multiple channels are usually on multiple controllers or even on different internal nodes. A traditional message can only be transferred through one channel, which will seriously waste the costly off-chiplet bandwidth.

III. INTERCONNECTION NETWORKS OF CHIPLETS

A. On-Chiplet Network

Limited by the layout and wiring capabilities of the chip, most on-chip interconnection networks have a flat topology. We adopt the most commonly used 2D-mesh as the topology of the on-chip network. As Fig. 3(a) shows, we classify all the nodes of the 2D-mesh into two categories: edge nodes and internal nodes. Edge nodes, also called interface nodes, are routers attached with external interfaces such as chiplet-to-chiplet interfaces or memory controllers; internal nodes, also called core nodes, are routers attached with functional modules such as CPU cores and AI cores. The microarchitecture of the router is the typical virtual channel router [49], which contains a few separate buffers at each input port. The switching technology of the router is based on VCT, which is widely adopted in the on-chip network. The flow control policy is not strictly constrained. We will provide a credit-based advanced flow control approach in Section IV-D. Optional hardware support includes a modifiable packet header, configurable routing tables, and software-defined virtual channels. Such support is not indispensable but can improve routing flexibility and performance.

In our method, each node of the on-chip network has a label for routing. We show a practical labeling example in Fig. 3(b) that works well with our routing algorithm. For all core nodes, the labels are the same as traditional 2D-mesh; and for edges nodes, they form a negative label ring.

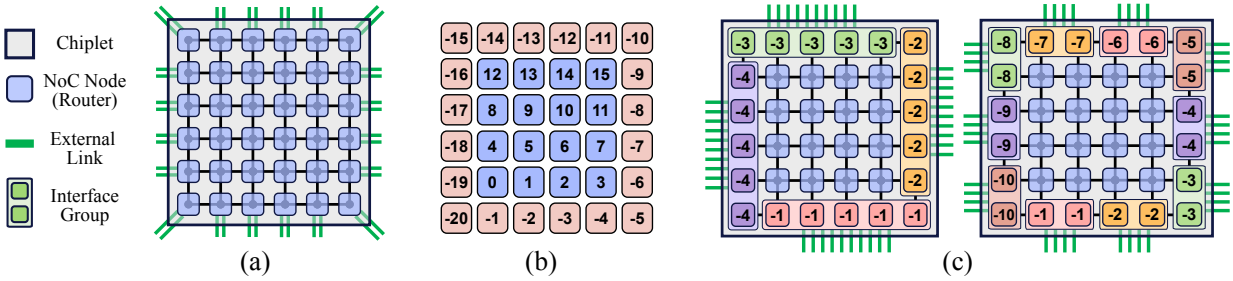

Fig. 3. Network on the chiplet and interface grouping. (a) 36 (6×6 2D-mesh) NoC nodes on the chiplet based on the typical virtual channel microarchitecture; (b) The node labels of the core nodes and the edge nodes; (c) Two interface grouping schemes for the 6×6 2D-mesh: radix-4 and radix-10.

With labels like this, any message from the core nodes can traverse all interfaces along a path with reduced labels; At the same time, any message from interface nodes can traverse all core nodes along a path with increased labels. Labeling-based routing algorithms are widely used, but when connecting many chiplets together in various topologies, the old routing algorithm may not work anymore. In Section IV, we present algorithms that implement deadlock-free high-performance routing in high-radix chiplet interconnection networks with such labels

B. Interface Grouping

Before interconnecting, we must make the necessary abstraction of the chiplet. Although each node of the 2D-mesh NoC is radix-4 (except the local port), the whole chiplet is a high-radix node with numerous external interfaces. Nevertheless, if the chiplet-to-chiplet interface of each edge node is regarded as an independent output port of the chiplet node, the radix of the chiplet node could be too high, and the bandwidth of each channel would be insufficient for inter-chiplet communication. Therefore, we adopt a method called *interface grouping* to cluster the adjacent edge nodes into several groups. For all physical interfaces in a group, we treat them as a single software-defined interface and always connect them to the same size interface on another chiplet. As Fig. 3(c) shows, a 6×6 2D-mesh has 20 edge nodes, and these nodes can be clustered into 4 or 10 groups. After interface grouping, chiplets can be regarded as large nodes with flexible radix. In this way, we can flexibly adjust the radix of the chiplet node and the bandwidth on each edge. Such a grouping method is not based on hardware but is software-defined, so it does not change the original NoC architecture.

However, the software-defined grouping approach can probably lead to bandwidth underutilization. Although the two chiplets are connected through multiple physical interfaces after grouping, each packet is transmitted along only one of the channels. To make matters worse, the link bandwidth inter-chiplet is usually less than that intra-chiplet. This makes the overall system performance limited by the chiplet-to-chiplet link. Inspired by the multi-channel memory system, we propose a method called *network interleaving* to make full use of the interface bandwidth. More details will be given in Section V.

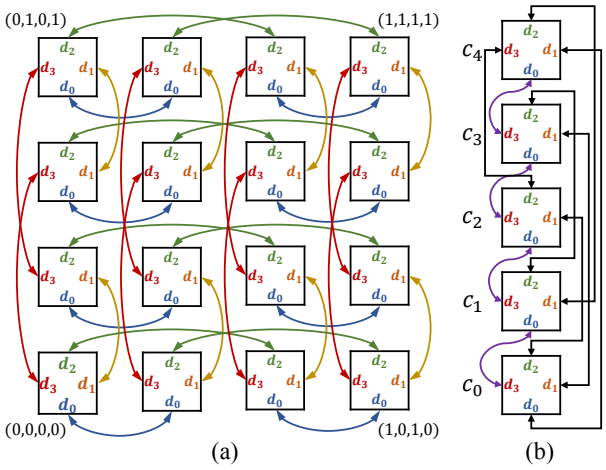

Fig. 4. Interconnection topologies based on the same chiplet: (a) hypercube and (b) dragonfly (fully connected).

C. Chiplets Interconnections

After interface grouping, each chiplet can be regarded as a high-radix node. Therefore, chiplets can support topologies with shorter diameters. Unlike ports of the ordinary router, interface groups of the chiplet nodes are not equivalent, and different connections will significantly impact routing and performance. There are some perceptual observations; For example, if we can connect two interface nodes of the same label, then MFR can be smoothly extended from one chiplet to another; When routing through the intermediate chiplets, the closer the inbound and outbound interfaces are to each other, the lower the communication latency. Therefore, we need to design our connections carefully. We take three typical high-radix topologies as examples.

As Fig. 4(a) shows, chiplets can be connected into a hypercube of 2^n nodes by clustering interfaces into n groups. The detail is present in Algorithm 1. The dimension of the interface group grows along the edge ring, and each chiplet-to-chiplet link connects ports of the same dimension, ensuring the labels' consistency across chiplets to facilitate routing. In other words, the relative position of the message remains the same after it is transmitted across the chiplet. Compared with 2D-mesh, the diameter of the hypercube reduces from $2(\sqrt{N}-1)$ to $\log_2 N$, and it doesn't waste ports at the edge.

Algorithm 1 CONNECTING CHIPLETS INTO HYPERCUBES

Input:

2^n chiplets,

n interface groups for each chiplet;

1: **for** $i = 0$ to $2^n - 1$ **do**

2: Coordinate of the i^{th} chiplet $C_i = (i\,\%2, (i/2)\%2, \cdots, (i/2^{n-1})\%2)$;

3: **for** $j = 0$ to $n - 1$ **do**

4: Connect the output ports of the j^{th} interface group of C_i to the input ports of the j^{th} interface group of chiplet $C_{i,j} = (i\,\%2, \cdots, \mathbf{1} - (\mathbf{i/2^j})\%\mathbf{2}, \cdots, i/2^{n-1})$;

5: **end for**

6: **end for**

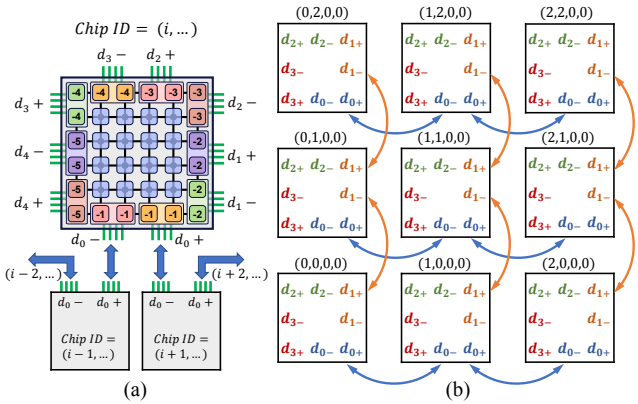

Fig. 5. Chiplets interconnection topologies: (a) connection and label details of the 5D-mesh; (b) connection overview of the d_0-d_1 plane of the 4D-mesh.

As Fig. 4(b) shows, chiplets can also be connected into a dragonfly (fully connected) network of $n + 1$ nodes by clustering interfaces into n groups. The connection method is derived from [42] and [43], which fits the MFR algorithm well. Compared with 2D-mesh, the diameter of the dragonfly reduces from $2(\sqrt{N} - 1)$ to 1, and it also doesn't waste ports at the edge.

Another typical topology is nD-mesh. As Fig. 5 shows, chiplets can be connected into an nD-mesh by clustering interfaces into $2n$ groups. The dimension of the interface group grows along the edge ring; each dimension has two directions: negative ($d-$) and positive ($d+$). Chiplet-to-chiplet link connects the positive and negative interfaces of adjacent chiplets in the same dimension. That is, messages that travel across chiplets fall in different directions of the same dimension. Compared with 2D-mesh, the diameter reduces from $2(\sqrt{N} - 1)$ to $n(\sqrt[n]{N} - 1)$.

In addition to supporting the above homogeneous topologies, irregular topologies can also be supported when only a single chiplet is used. As Fig. 6 shows, due to the scalability of 2D-mesh NoC, we can use multiple chiplets to form a larger 2D-mesh. After interface re-grouping, heterogeneous networks such as the tree and even irregular networks can be connected.

Although we can interconnect chiplets in various ways, it is a long way from a high-performance network. On these complex and high-radix networks, we need to design efficient and deadlock-free routing algorithms. More details about routing will be discussed in the next section.

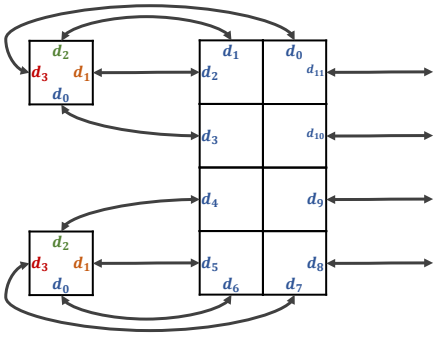

Fig. 6. Chiplet interconnection topologies: tree (irregular topology). Multiple 2D-mesh can be built into a larger 2D-mesh, which leads to a higher-radix chiplet node.

IV. ROUTING DESIGN

This section presents our methodology for designing a deadlock-free routing algorithm for interconnection networks of chiplets. We will first give the overall framework of the routing algorithm and then give a concrete example of the chiplet hypercube network. Finally, we will introduce an advanced flow control technique that simplifies routing. As we described in Section II, we only need to design a deadlock-free subnetwork. The channels in this section refer to these escape channels, while those not mentioned can be used fully adaptively. Before describing our methodology, we define some necessary concepts.

Definition 1. *A physical or virtual channel from a larger-label node to a smaller-label node is called a minus channel; A physical or virtual channel from a smaller-label node to a larger-label node is called a plus channel; A physical or virtual channel connecting nodes of the same label is called an equal channel.*

Definition 2. *A node inside the 2D-mesh-NoC is called a Core; a node on the edge of the 2D-mesh-NoC is called an Interface (IF).*

Definition 3. *Minus-first Routing (MFR) is a routing algorithm that restricts turning to a minus channel while the current channel direction is plus. In other words, a message is delivered across a few minus hops first and then across a few plus hops to the destination. Paths that follow the routing rule are called minus-first paths.*

Definition 4. *A packet is delivered and kept in the next input buffer as a safe packet if it has a minus-first path from the current channel to the destination; otherwise, the packet is unsafe.*

Definition 5. *A equal label group is a group of connected nodes with the same label.*

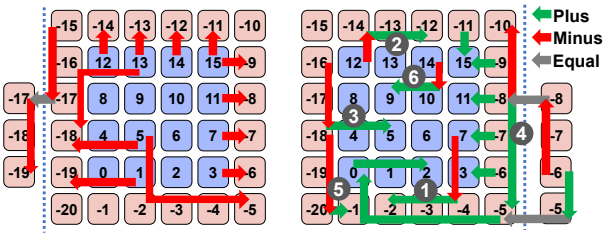

Fig. 7. Overview for three kinds of channels: plus, minus, and equal.

Algorithm 2 MFR(C,P) AMONG CHIPLETS

Input:
current node C,
packet P,

1: $Offset$ = different dimensions of chiplet coordinates.
2: assert($Offset$! = $\emptyset$)
3: B = IF node of dimension $d_B \in Offset$ that need to transfer first
4: **if** C == B **then**
5: transfer to adjacent chiplet through outward channel
6: **else**
7: transfer to B based on MFR(C,P) WITHIN CHIPLET
8: **end if**

A. MFR Algorithm

The general idea of designing the MFR algorithm is to find a "minus-first" path for any node pair based on the labels. As we described in Section III-A, the labels of the cores are the normal label of the 2D-mesh and the labels of the IFs are a ring of negative. Therefore, we can give an overview of the channel properties. As shown in Fig. 7, red arrows are minus channels, and green arrows are plus channels. In particular, the turns from minus to plus can be summarized into six situations: ①From a minus channel (also the negative channel in 2D-mesh) turn to a plus channel along the IF ring; ②From a minus channel (original positive channel in 2D-mesh) turn to a plus channel along the IF ring; ③From a minus channel along the IF ring turn to a plus channel pointing to the core; ④From a minus channel along one chiplet IF ring turn to a plus channel of another chiplet IF ring; ⑤From a minus channel along the IF ring to the end and then turn to the plus channel (-20 to -1). ⑥From a negative channel turn to a positive channel (among cores).

Routing across chiplets is relatively easy to describe because inter-chiplet channels always go from IF nodes to IF nodes. As Algorithm 2 shows, the routing algorithm only needs to determine which IF to leave from; if the message arrives, transfer to the adjacent chiplet through the outward channel; if it does not arrive, call $MFR(C, P)$ WITHIN CHIPLET to reach the IF.

As Algorithm 3 shows, the in-chiplet routing can be divided into four types: *core-to-core*, *IF-to-IF*, *IF-to-core*, and *core-to-IF*. These algorithms are invoked in stages for routing any message. For intra-chiplet messages, the minus-first rule is easy to maintain. However, for inter-chiplet messages, the path needs to be carefully planned. For example, for a core-to-core inter-chip message, we need to call the CORE_TO_IF(C,P) first to send the message to the outward interface by minus-only path. All subsequent cross-chip transmissions are on the interface ring based on IF_TO_IF(C,P) until the target chiplet is reached. Then, IF_TO_CORE(C,P) is called after passing through the chiplet-to-chiplet link. Since channels from the interface into the core mesh are plus channels, we must send the message to an appropriate interface, such as interface -1 or -19 in Fig. 7, so that the message doesn't need to go through a minus channel later. There is a free chance for a message to turn from the minus direction to the plus direction as it travels through the interface rings. However, more redirects must be introduced with extra virtual channels or other flow control technologies. For most topologies that can adopt dimensional-order routings, such as nD-mesh and hypercube, our algorithm can be perfectly adapted.

[1]Equality implies that adjacent nodes are the target core node

Algorithm 3 MFR(C,P) WITHIN CHIPLET

Input:
current node C,
packet P,

Function: CORE_TO_CORE(C,P)
transfer based on NEGTIVE_FIRST_ROUTING(C,D).

Function: IF_TO_IF(C,P)
transfer along the IF ring, out of the chiplet through the external link if necessary; only one turn from minus to plus channels is allowed

Function: IF_TO_CORE(C,P)
transfer along the IF ring until
the label of adjacent core node $\leq$ destination[1], then transfer based on CORE_TO_CORE(C,P) by only plus channels.

Function: CORE_TO_IF(C,P)
transfer to an IF node by only minus channels, then transfer based on IF_TO_IF(C,P).

B. Deadlock Issues of Equal Channels

In Section III-B, we propose an interface grouping method that abstracts several edge nodes as one external interface. If we assign the same label to all the nodes in the group during routing design, it may lead to potential deadlocks. This is because the channel in the group is a new kind of channel: *equal channel*. *Equal channel* also occurs between chiplets if we connect interfaces of the same label of two chiplets. The channel dependency chains *plus-to-equal-to-minus* and *equal-to-equal* may result in deadlocks. For example, Fig. 8 demonstrates a deadlock configuration of the nD-mesh topology. The dependency circle forms among the interface groups of the

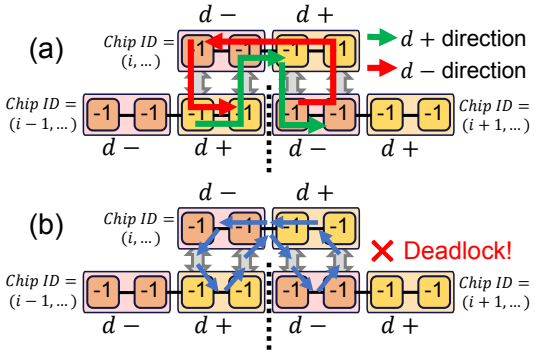

Fig. 8. Deaklock inside the interface groups of the nD-mesh topology. In this dimension, packets moving in two different directions form a channel dependency circle.

same label in the same dimension. Therefore, we have to solve this problem.

One straightforward way is to label each physical interface individually. However, the interface group can no longer be treated as an entire node, significantly increasing the difficulty of routing design among chiplets. Moreover, once interfaces cannot be grouped according to the dimensions of the chiplet topology, many algorithms and conclusions based on dimensional order routing are no longer applicable. Therefore, we improve MFR to solve the deadlock problem while keeping the group's unified label. The revision of the MFR is shown in Theorem 1.

Theorem 1. *The MFR algorithm with equal channels is deadlock-free if the following conditions are met:*

1. *A packet that has passed through the plus channel is prohibited from turning to the minus channel;*
2. *In the equal label groups, virtual channels are used to separate packets entering along the plus-channel (if any) from other packets;*
3. *There is no deadlock in the equal label groups.*

Proof. Suppose there is a channel dependency circle. As there is no deadlock in the *equal label group* according to condition 3, at least one channel of the circle is not equal. So there must be a chain of dependency going from a plus channel to a minus channel. As turning from a plus channel directly to a minus channel is prohibited by the MFR, there must be at least one equal channel in the middle of the chain, which means that the chain must go through at least one *equal label group*. According to condition 2, packets entering the group along the plus channel will go to another virtual sub-network, and according to condition 1, packets in this virtual sub-network cannot turn to the minus channel as they leave. Therefore, the channel dependency circle cannot exist; That is, the MFR algorithm with *equal channels* is deadlock-free. □

Based on this theorem, we can solve cross-group deadlocks by adding virtual channels at the interface nodes. For deadlocks within the *equal label groups*, as dependencies are isolated from the outside, we just need to design the routing

Algorithm 4 MFR(C,P) FOR HYPERCUBES

Input:
current node C,
packet P;

1: $Offset$ = different dimensions of chiplet coordinates. each dimension is associated with an IF group;
2: **while** $Offset\ != \emptyset$ **do**
3: transfer to an IF node B ($d_B = \min\{Offset\}$ with the largest label) based on IF_TO_IF(C,P) or CORE_TO_IF(C,P) by only minus channel;
4: transfer to adjacent chiplet through outward channel;
5: **end while**
6: **if** $P.destination$ is IF node **then**
7: transfer based on $IF\_to\_IF(C, P)$;
8: **else if** $P.destination$ is core node **then**
9: transfer based on $IF\_to\_Core(C, P)$;
10: **end if**

individually. For the nD-mesh topology, before packets reach the destination chiplet in a dimensional order along the minus path, they do not need to go through any plus channel. Hence, virtual channels in condition 2 is not necessary. For deadlock in the equal label groups, one virtual channel can be used to separate the packets in $d+$ and $d-$ directions. Therefore, global deadlock-free routing can be realized for nD-mesh by adding one virtual channel. Since the hypercube and dragonfly networks have neither cross-equal-group deadlock nor intra-equal-group deadlock, global deadlock-free routing can be achieved without virtual channels.

C. Case Study: Hypercube

The hypercube is a topology with simple structure and short diameter. As it is very popular and easy to describe, we present the MFR on hypercube as a specific example. In Section III-C, we have already shown how to build a hypercube with 2D-mesh chiplets. Dimensions grow along the ring, and interfaces of the same dimension (label) are connected.

Routing detail is present in Algorithm 4. Each chiplet has its coordinates in the hypercube, from which we can calculate the $Offset$, which are the dimensions that need routing. The interface nodes of these dimensions are distributed along the interface ring, and a message can traverse all these dimensions by going around the interface ring through a minus-only path. Once a message reaches the target chiplet, if the destination node is an interface node, it is sent directly along the ring by minus or plus direction; the minus-first principle has been guaranteed. If the destination node is a core node, the message continues along the interface ring until either the neighboring node is the destination node or the X and Y coordinates of the current interface in the 2D-mesh-NoC are smaller than the destination node. Then, the message can easily take a plus-only path from the interface to the destination node.

Unlike the nD-mesh topology, the hypercube has only one direction in each dimension. Therefore, for the hypercube, messages do not go back as they traverse each dimension on

Algorithm 5 $VC\_Allocation(f, s)$

Input:
number of *available* VCs a,
number of *safe* packets s;

if $a \geq 2$ **then**
  return true;
else if $a = 1$ and $s \geq 1$ **then**
  return true;
else if $a = 1$ and $s = 0$ **then**
  if the packet is *safe* at the next router **then**
    return true;
  else
    return false;
  end if
else if $a = 0$ **then**
  return false;
end if

the interface ring. In other words, messages keep going in the minus direction until finishing the last dimension. There are neither cross-equal-group nor intra-equal-group deadlocks for hypercubes. Therefore, global deadlock-free routing can be achieved without any virtual channels for hypercube networks.

D. MFR with Safe/Unsafe Flow Control

For various topologies, especially for irregular networks, it is tedious to find a minus-first path for any two pairs of communication nodes. Moreover, in order to comply with the minus-first principle, many routing paths are not the shortest paths. To overcome these limitations, we provide a general method to achieve convenient deadlock-free routing when only at least two unclassified buffers are required. The method is based on flow control and was first implemented on 2D-mesh networks to balanced buffer utilization [50] [51].

The definitions of *safe* and *unsafe* packets are shown in Definition 4. By definition, we know that all the *safe* packets have a path that conforms to the original routing algorithm to reach the destination. As shown in Algorithm 5, *safe/unsafe* flow control policy will check the status of the target router in the VC allocation stage. If there is more than one available channel or an available channel with at least one *safe* packet, the packet is allowed to move. Or, if the packet is *safe* at the next input port, it is also allowed to move. Otherwise, the packet is prohibited from delivering.

Since each input port must have an empty channel or a *safe* packet, and by Definition 3, 4, and Theorem 1 all *safe* packets cannot form a deadlock circle. Therefore, a *safe* packet must not be permanently blocked. As for *unsafe* packets, a deadlock configuration requires many *unsafe* packets to form a channel dependency circle. In this case, as long as any of these packets can be sent to an adjacent node and become a *safe* packet, deadlocks are eliminated. The following three conditions must be met for a deadlock to occur:

1. Packets form a dependency circle;

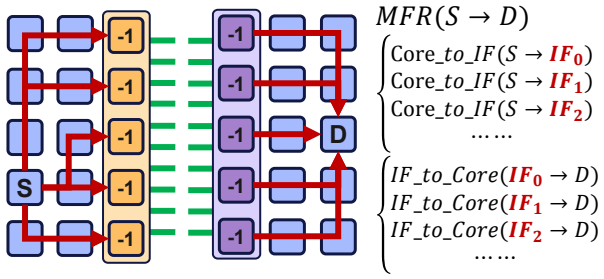

Fig. 9. Network Interleaving is to evenly distribute the cross-chip communication on the whole interface group. It is implemented by changing the target interface when calling CORE_TO_IF(C,P).

2. None of these packets has a minus-first path from the current channel to the destination;
3. None of these packets has a minus-first path from an adjacent channel to the destination.

This is a stringent condition that is impossible to satisfy in most cases. The *safe/unsafe* flow control relaxes some deadlock restrictions but brings excellent routing design convenience. With this policy, all packets do not have to follow the minus-first path but the shortest path. Another advantage of this approach is that it even allows some pairs of communication nodes to have no minus-first path between them. This flow control technique is of great help in extending the MFR algorithm to more general network architectures.

V. Network Interleaving

In Section III-B, we group several interface nodes into one abstract interface. Since the grouping is software-defined, a traditional message can only come out from one physical channel. This resulted in a significant loss of bandwidth gains from grouping. Therefore, we propose a new method called *network interleaving* to improve inter-chiplet communication efficiency.

As shown in Fig. 9, when the source node sends a message to the destination node on another chip, we distribute the traffic evenly across the entire abstract interface. In this way, the inter-chip communication bandwidth can be fully utilized. The implementation of interleaved routing is very straightforward, requiring only changing the target interface in calling algorithm CORE_TO_IF(C,P). The hardware needs to modify the packet header during message packaging (add different tags), and the routers will send the packets to different interface nodes according to the tags.

Just like memory interleaving, network interleaving has different granularity. Different interleaved granularity has different performance characteristics and different requirements on hardware. We will introduce two typical interleaving styles in detail.

A. Fine-grained Interleaving

The *fine-grained interleaving* is to distribute the packets of a message to multiple physical interfaces of an abstract interface one by one. It alternately assigns different *interleaving-tags* to the packet headers and sends packets to different physical

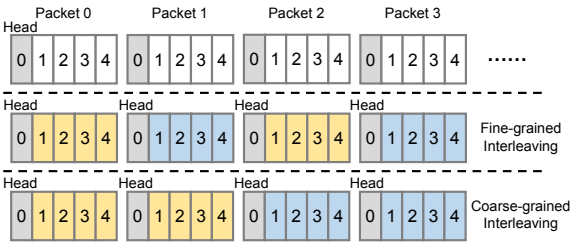

Fig. 10. Two different interleaving granularity.

interfaces. This fine-grained interleaving has the highest bandwidth utilization because it works even under light network load and is most evenly distributed spatially across interface nodes.

However, fine-grained interleaving has drawbacks. The packets of a message must be individually packaged, and the routing calculation burden increases. When transmitting a single message, all physical interfaces are active, increasing the total power consumption. Moreover, fine-grained technology can introduce more serious *out-of-order* problems

B. Coarse-grained Interleaving

The *coarse-grained interleaving* is to distribute packets in a coarser-grained way. As shown in Fig. 10, for every few packets or even every message, the router assigns a different *interleaving-tag*. Compared with fine-grained interleaving, coarse-grained interleaving benefits under heavy network load. For a single message, there are only one or a few physical channels of bandwidth available.

Nevertheless, coarse-grained interleaving is implement-friendly. Based on the traditional router microarchitecture, little additional hardware is required to implement message-level interleaving. Under low network load, the coarse-grained interleaving reduces the power consumption of physical interfaces. However, there is still a problem of *out-of-order*, which is not discussed in this article.

VI. Evaluation Methodology

A. Simulation Model & Parameter Setting

To evaluate the network performance, we designed a chiplet-specific cycle-accurate C++ simulator. The router model of the simulator is based on the typical virtual channel router [49]. The switching technology is based on virtual cut-through (VCT). It contains a pipeline of 4 stages: routing, VC allocation, switch allocation, and transmission. The crossbar is preemptively scheduled, and starvation is avoided through the *first-come-first-serve* strategy. Flow control is based on credit, and each output port can know the occupation of the corresponding input buffer. The cross-chiplet VC allocation and cross-chiplet message transmission stages consume more clock cycles. Compared with other general-purpose network simulators such as *booksim* [52], Our simulator allows non-uniform link configurations to reflect different properties inside and outside the chiplet. Each port's bandwidth, latency, and buffer size can be configured individually. It can also introduce longer delays for the cross-chiplet pipeline stages.

TABLE II
Parameters Setup

Parameter	Value
Flit Width	32 bits
Packet Length	32 flits
Input Buffer Size	1024 bits (32 flits) for internal buffers 2048 bits (64 flits) for interface buffers
Virtual Channel Number	2 channels/port
On-chip Link Bandwidth	128 bits/cycle (4 flits/cycle)
Off-chip Link Bandwidth	64 bits/cycle (2 flits/cycle)
Pipeline	4 stages; 1 cycle/stage
Intra-Chip Link Extra Delay	5 cycle
Simulation Time	6000 cycles (1000 for warming up)

The simulator's default parameters used in the following experiments are shown in Table II. We use *Duato's protocol* [39] based adaptive negative-first routing (NFR) on 2D-mesh as the baseline. This is reasonable and fair in terms of hardware and routing algorithm. The performance differences are mainly reflected by the chiplet interconnection method, cross-chiplet routing, and interleaving, which is the main methodology of this paper.

B. Evaluation on Traffic Patterns

We evaluate our method on 6 different traffic patterns. Uniform (random) traffic can be described by a traffic matrix with all entries are $\lambda_{sd} = 1/N$. Uniform-hotspot is a sparse matrix with traffic only between random 10% of the node pairs. The other 4 patterns are *permutation traffic*, in which all traffic from each source is directed to one destination. These patterns are based on interconnection patterns that arise in particular applications. For example, matrix transpose induces the transpose pattern, and fast Fourier transform (FFT) causes the shuffle pattern [53]. Node labels are global, so there is both internal (within chiplet) and global (between chiplets) traffic. The scale of the system is 64 4×4-2D-mesh chiplets. The topologies (grouping approaches) are 2D-mesh (8×8, as the baseline), 3D-mesh ($4 \times 4 \times 4$, radix-6 grouping), and hypercube (2^6, radix-6 grouping).

As shown in Fig. 11, for all traffic patterns, our method has a significant latency advantage over traditional adaptive routing in 2D-mesh. Our MFR on hypercube has a saturation injection rate of more than double at all traffic patterns. 3D-mesh has an advantage at some patterns such as *Bit-reverse*.

The evaluation results show that our method can provide a better interconnection scheme than 2D-mesh for various traffic loads. Our methodology has a greater advantage under traffic with mass global (cross-chiplets) communications.

VII. Explorations

A. Exploration on System Scales

Scale is the most important factor in the design of interconnection networks. We evaluate 4 different scales on three topologies. The chiplet-to-chiplet link bandwidth is set to 64 bits/cycle (2 flits/cycle), and coarse-grained interleaving is enabled.

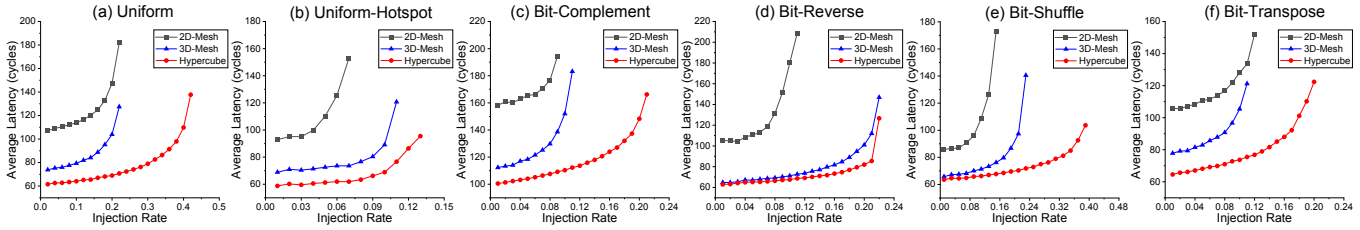

Fig. 11. Evaluations on different traffic patterns. (a) Uniform (Random): $\lambda_{sd} = 1/N$; (b) Uniform-hotspot: random 10% node pairs; (c) Bit-complement: $d_i = \neg s_i$; (d) Bit-reverse: $d_i = s_{b-i-1}$; (e) Bit-shuffle: $d_i = s_{(i-1) \bmod b}$; (f) Bit-transpose: $d_i = s_{(i+b/2) \bmod b}$.

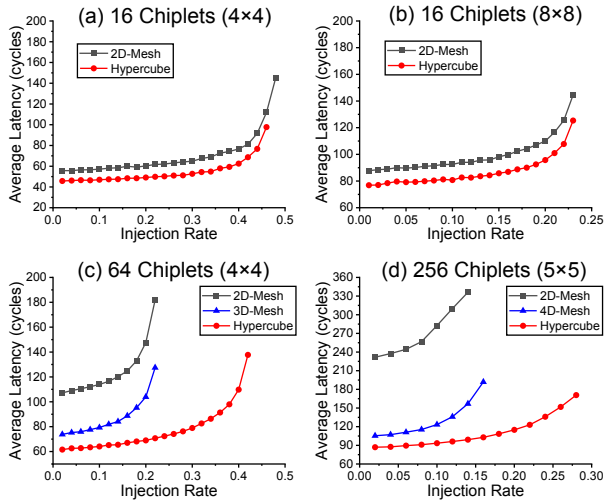

Fig. 12. Average latency evaluation results on topologies and scales (NoC scales) under uniform traffic. Off-chip link bandwidth is 64 bits/cycle.

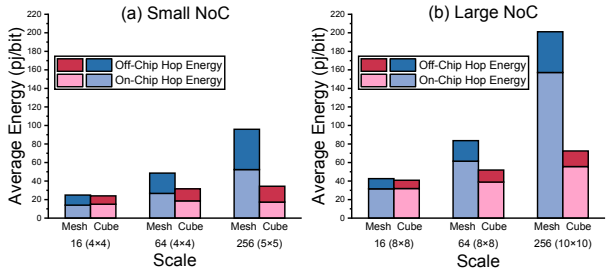

Fig. 13. Energy consumption estimation on different scales. 2D-mesh vs. hypercube under different chiplet numbers and NoC scales.

As shown in Fig. 12(a), when the scale of the on-chiplet network is raised to 8×8 in Fig. 12(b), the reduction in latency is smaller because the percentage of in-chiplet transmission latency increases. Then, we increase the number of chiplets to 64 and 256 in Fig. 12(c)(d). nD-mesh and 2D-mesh have similar saturation injection rates, but MFR on nD-mesh still has significant latency advantages: ∼35% for 3D-mesh and ∼55% for 4D-mesh. Furthermore, our MFR on hypercube topology is much better than the baseline in both latency and saturation injection rate. Up to 2× saturation injection rate is achieved with latency smaller than 50%.

We also estimate the average energy consumption based on 130nm technology on different scales. The on-chip energy consumption is based on the model presented by [54]. Under $2mm$ wire length, the dynamic energy required to transport a message between two nodes is estimated by 0.98 pj/bit per router and 0.63 pj/bit per link. The overhead of the off-chip link is estimated by 2.4 pj/bit according to the recent work [55]. Based on these data, we evaluate the average energy cost of message delivery for 2D-mesh and hypercube networks at different scales. The evaluation results are shown in Fig. 13. For the power consumption of both on-chip and off-chip transmission, our method has obvious advantages, especially for large-scale networks. For the 256-chiplets-hypercube, the power consumption is reduced by ∼60% compared with 2D-mesh.

In general, our methodology works better on more chiplets and is not sensitive to the scale of on-chip networks. The high-radix topology dramatically reduces the network diameter on a large scale compared with the 2D-mesh. A shorter chiplet network diameter not only reduces the number of hops across chiplets but also reduces the number of hops within passing chiplets. Therefore, the average hop number decreases are reflected in lower latency and power consumption.

B. Exploration on Chiplet-to-Chiplet Link Configuration

Another factor that can have a significant impact on the performance is the chiplet-to-chiplet link. We evaluate four different bandwidth configurations on 64 4×4-2D-mesh-based chiplets. Coarse-grained interleaving is used, and the chiplet-to-chiplet link bandwidth is set to 1/4×, 1/2×, 1×, and 2× of the on-chip link bandwidth.

As shown in Fig. 14(a), inadequate bandwidth greatly limits the network performance. Both 2D and 3D meshes quickly reach the saturation point, but the saturation injection rate of the hypercube network is still twice that of the mesh. When link bandwidth is raised to 64 bits/cycle (2 flits/cycle) in Fig. 14(b), the overall performance comparison does not change. The hypercube network has twice the saturation injection rate and half the average latency. However, with the further increase in link bandwidth, performance improvements in high-radix networks have stalled. As shown in Fig. 14(c)(d), the high-radix network still has the low latency advantage, but the saturation injection rate is close to or even worse than the 2D-mesh. This is because the bottleneck of the network

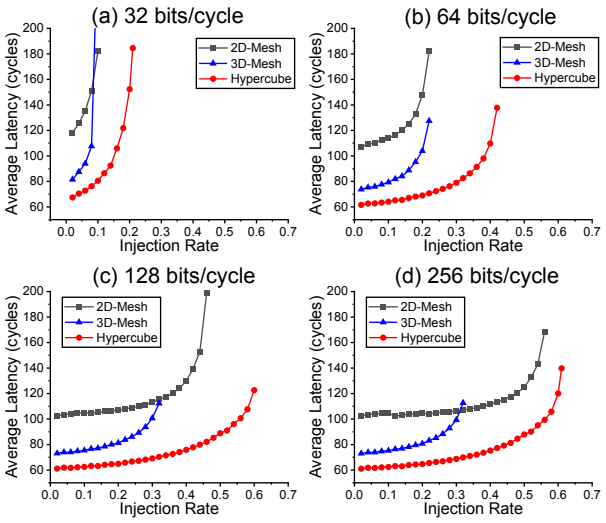

Fig. 14. Average latency evaluation results on chiplet-to-chiplet link bandwidth under uniform traffic. The chiplet number is 64, and the NoC scale is 4×4.

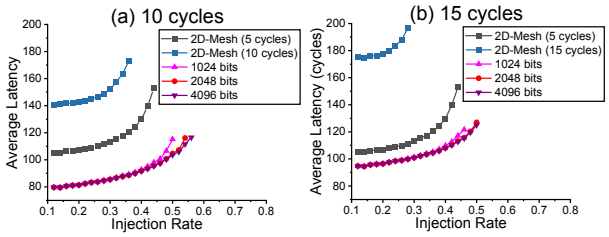

Fig. 15. Average latency evaluation results on chiplet-to-chiplet link latency and buffer size under uniform traffic. The chiplet number is 64, and the NoC scale is 4×4. Off-chip link bandwidth is 64 bits/cycle. The link buffer size for the 2D-mesh is 2048 bits.

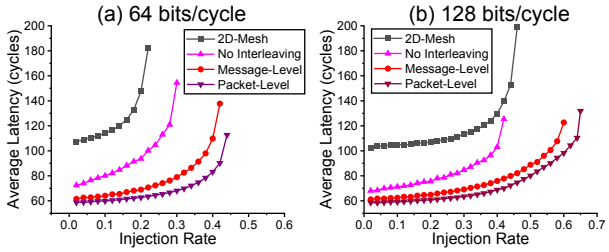

Fig. 16. Average latency evaluation results on interleaving under uniform traffic. T chiplet number is 64, and NoC scale is 4×4. The topology of the experimental group is the hypercube.

is no longer cross-chiplet communication but intra-chiplet communication.

We also evaluate the impact of the chiplet-to-chiplet link latency and the link buffer size. As 2D-mesh allows for short-reach links with lower latency, we keep the configuration of 5 cycles delay and 2048 bits buffer size for 2D-mesh. Hypercube topology is used in the experimental groups, and different link configurations are evaluated. The results are shown in Fig. 15. With the increase of the link delay, the average latency of message delivery increases rapidly. The buffer size of the link has little impact on performance. 64-hypercube with 15-cycles link delay still has better performance than 64-2D-mesh with 5-cycles link delay, but the lead is smaller.

In summary, our method has a significant positive effect on bandwidth-constrained multi-chiplet systems. By reducing the network diameter and average hop count, our method reduces the bandwidth usage of in-transit packets for on-chip and off-chip links, thus improving the network performance. As for the system with abundant inter-chiplet link bandwidth, our method can still significantly reduce the average latency, but the injection rate bottleneck becomes the on-chip network. The link delay impacts the system more than the buffer size. Although 2D-Mesh can use the interface with shorter reach and delay, it still lacks the performance and flexibility of our approach.

C. Exploration on Interleaving

The above experiments all enabled interleaving. We also evaluated configurations with no interleaving and different interleaving granularity on 64 4×4-2D-mesh-based chiplets. The chiplet-to-chiplet link latency is 5-cycles. Configuration (a) is 64 bits/cycle link bandwidth and Configuration (b) is 128 bits/cycle link bandwidth.

As shown in Fig. 16, interleaving provides a considerable performance boost. Without interleaving, there are ∼15% increase in average latency and ∼20% decrease in saturation injection rate. Compared with message-level interleaving, packet-level interleaving brings ∼10% increase in saturation injection rate and less than 5% decrease in average latency. In the configuration of 64 bits/cycle link bandwidth, the improvement of interleaving is more evident. This is because, in bandwidth-constrained scenarios, the improvement of bandwidth utilization brings more significant performance improvement.

In general, interleaving is important for networks of chiplets, especially in bandwidth-constrained scenarios. Fine-grained interleaving performs better than coarse-grained interleaving, but this is achieved with more complex hardware.

VIII. Summary

Seeing the limitations of the existing chiplet-based networks, we have proposed a methodology to design more efficient and scalable multi-chiplet interconnect networks. Based on the new software-defined interface grouping method, we have presented a specification for connecting 2D-mesh-NoC-based chiplets into high-radix networks. A deadlock-free adaptive routing algorithm based on MFR has been proposed on these networks. To overcome the potential limitations of the new approach, *safe/unsafe* flow control and *network interleaving* methods have been proposed. A chiplet-specific cycle-accurate C++ simulator is used to evaluate the new methodology. We have done many explorations under a variety of network configurations. The evaluation and exploration results demonstrate our approach's high-performance and flexibility.

REFERENCES

[1] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. Valencia, Spain: IEEE, Jun. 2021, pp. 57–70.

[2] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony, "2.2 amd chiplet architecture for high-performance server and desktop products," in *2020 IEEE International Solid State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2020, pp. 44–45.

[3] W. Gomes, A. Koker, P. Stover, D. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, F. O'Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, and R. Koduri, "Ponte vecchio: A multi-tile 3d stacked processor for exascale computing," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2022, pp. 42–44.

[4] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, C. T. Gray, B. Khailany, and S. W. Keckler, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. Columbus OH USA: ACM, Oct. 2019, pp. 14–27.

[5] F. Zaruba, F. Schuiki, and L. Benini, "Manticore: A 4096-core risc-v chiplet architecture for ultraefficient floating-point computing," *IEEE Micro*, vol. 41, no. 2, pp. 36–42, Mar. 2021.

[6] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. S. Emer, C. T. Gray, S. W. Keckler, and B. Khailany, "A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.

[7] B. Chang, R. Kurian, D. Williams, and E. Quinnell, "Dojo: Supercompute system scaling for ml training," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. Cupertino, CA, USA: IEEE, Aug. 2022, pp. 1–45.

[8] E. Talpes, D. Williams, and D. D. Sarma, "Dojo: The microarchitecture of tesla's exa-scale computer," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. Cupertino, CA, USA: IEEE, Aug. 2022, pp. 1–28.

[9] S. Lie, "Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning : Cerebras systems," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. Cupertino, CA, USA: IEEE, Aug. 2022, pp. 1–34.

[10] G. Lauterbach, "The path to successful wafer-scale integration: The cerebras story," *IEEE Micro*, vol. 41, no. 6, pp. 52–57, Nov. 2021.

[11] Cerebras, "Wafer-scale deep learning," in *2019 IEEE Hot Chips 31 Symposium (HCS)*. Cupertino, CA, USA: IEEE, Aug. 2019, pp. 1–31.

[12] S. Lie, "Multi-million core, multi-wafer ai cluster," in *2021 IEEE Hot Chips 33 Symposium (HCS)*. Palo Alto, CA, USA: IEEE, Aug. 2021, pp. 1–41.

[13] "Designware die-to-die 112g usr/xsr phy & die-to-die controller," https://www.synopsys.com/dw/ipdir.php?ds=dwc_usr-xsr_phy.

[14] "Synopsys die-to-die IP," https://www.synopsys.com/designware-ip/interface-ip/die-to-die.html.

[15] P. K. Huang, C. Y. Lu, W. H. Wei, C. Chiu, K. C. Ting, C. Hu, C. Tsai, S. Y. Hou, W. C. Chiou, C. T. Wang, and D. Yu, "Wafer level system integration of the fifth generation cowos®-s with high performance si interposer at 2500 mm2," in *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*. San Diego, CA, USA: IEEE, Jun. 2021, pp. 101–104.

[16] Shin-Hua Chao and Chao-Fu Weng, "Fine line/space ic substrate made by selectively fully additive process," in *2016 International Conference on Electronics Packaging (ICEP)*. Hokkaido, Japan: IEEE, Apr. 2016, pp. 341–344.

[17] D. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, "Cost analysis and cost-driven ip reuse methodology for soc design based on 2.5d/3d integration," in *Proceedings of the 35th International Conference on Computer-Aided Design*, F. Liu, Ed. ACM, pp. 1–6.

[18] Y. Feng and K. Ma, "Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*. San Francisco California: ACM, Jul. 2022, pp. 121–126.

[19] M. Bedford Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, "Scalar operand networks: On-chip interconnect for ilp in partitioned architectures," in *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.* Anaheim, CA, USA: IEEE Comput. Soc, 2003, pp. 341–353.

[20] P. Gratz, C. Kim, R. McDonald, S. W. Keckler, and D. Burger, "Implementation and evaluation of on-chip network architectures," in *2006 International Conference on Computer Design*. San Jose, CA, USA: IEEE, Oct. 2006, pp. 477–484.

[21] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-tile 1.28tflops network-on-chip in 65nm cmos," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. San Francisco, CA, USA: IEEE, Feb. 2007, pp. 98–589.

[22] Tilera, "The tile processor™ architecture: Embedded multicore for networking and digital multimedia," in *2007 IEEE Hot Chips 19 Symposium (HCS)*. Stanford, CA, USA: IEEE, Aug. 2007, pp. 1–12.

[23] R. Vaidyanathan, J. L. Trahan, and S. Rai, "Introducing parallel and distributed computing concepts in digital logic," in *Topics in Parallel and Distributed Computing*. Elsevier, 2015, pp. 83–116.

[24] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. Shoaib Bin Altaf, N. Enright Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. Los Angeles, CA: IEEE, Jun. 2018, pp. 726–738.

[25] 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," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1928–1941, Nov. 2021.

[26] P. Vivet, E. Guthmuller, Y. Thonnart, G. Pillonnet, C. Fuguet, I. Miro-Panades, G. Moritz, J. Durupt, C. Bernard, D. Varreau, J. Pontes, S. Thuries, D. Coriat, M. Harrand, D. Dutoit, D. Lattard, L. Arnaud, J. Charbonnier, P. Coudrain, A. Garnier, F. Berger, A. Gueugnot, A. Greiner, Q. L. Meunier, A. Farcy, A. Arriordaz, S. Cheramy, and F. Clermidy, "Intact: A 96-core processor with six chiplets 3d-stacked on an active interposer with distributed interconnects and integrated power management," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 79–97, Jan. 2021.

[27] S. Pal, J. Liu, I. Alam, N. Cebry, H. Suhail, S. Bu, S. S. Iyer, S. Pamarti, R. Kumar, and P. Gupta, "Designing a 2048-chiplet, 14336-core waferscale processor," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. San Francisco, CA, USA: IEEE, Dec. 2021, pp. 1183–1188.

[28] D. Ignjatovic, D. W. Bailey, and L. Bajic, "The wormhole ai training processor," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2022, pp. 356–358.

[29] H. Farrokhbakht, H. Kao, K. Hasan, P. V. Gratz, T. Krishna, J. San Miguel, and N. E. Jerger, "Pitstop: Enabling a virtual network free network-on-chip," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. Seoul, Korea (South): IEEE, Feb. 2021, pp. 682–695.

[30] Y. Wu, L. Wang, X. Wang, J. Han, J. Zhu, H. Jiang, S. Yin, S. Wei, and L. Liu, "Upward packet popup for deadlock freedom in modular chiplet-based systems," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. Seoul, Korea, Republic of: IEEE, Apr. 2022, pp. 986–1000.

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

[32] G. H. Loh, S. Naffziger, and K. Lepak, "Understanding chiplets today to anticipate future integration opportunities and limits," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Grenoble, France: IEEE, Feb. 2021, pp. 142–145.

[33] J. Kim, S. Kundu, A. Balankutty, M. Beach, B. C. Kim, S. Kim, Y. Liu, S. K. Murthy, P. Wali, K. Yu, H. S. Kim, C.-c. Liu, D. Shin, A. Cohen, Y. Fan, and F. O'Mahony, "8.1 a 224gb/s dac-based pam-4 transmitter with 8-tap ffe in 10nm cmos," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2021, pp. 126–128.

[34] C. Minkenberg, R. Krishnaswamy, A. Zilkie, and D. Nelson, "Co-packaged datacenter optics: Opportunities and challenges," *IET Optoelectronics*, vol. 15, no. 2, pp. 77–91, Apr. 2021.

[35] "Jetson nano developer kit — nvidia developer," https://developer.nvidia.com/embedded/jetson-nano-developer-kit.
[36] "Nvidia dgx a100 : The universal system for ai infrastructure," https://www.nvidia.com/en-us/data-center/dgx-a100/.
[37] J. Choquette, E. Lee, R. Krashinsky, V. Balan, and B. Khailany, "3.2 the a100 datacenter gpu and ampere architecture," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2021, pp. 48–50.
[38] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.
[39] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*, rev. printing ed. San Francisco, CA: Morgan Kaufmann, 2003.
[40] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker, "Autonet: A high-speed, self-configuring local area network using point-to-point links," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1318–1335, Oct. 1991.
[41] D. Xiang and Y. Zhang, "Cost-effective power-aware core testing in nocs based on a new unicast-based multicast scheme," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 135–147, Jan. 2011.
[42] D. Xiang and X. Liu, "Deadlock-free broadcast routing in dragonfly networks without virtual channels," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2520–2532, Sep. 2016.
[43] D. Xiang, B. Li, and Y. Fu, "Fault-tolerant adaptive routing in dragonfly networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 259–271, Mar. 2019.
[44] C. Glass and L. Ni, "The turn model for adaptive routing," in *Proceedings the 19th Annual International Symposium on Computer Architecture*. Gold Coast, Australia: IEEE, 1992, pp. 278–287.
[45] G. J. Burnett and E. G. Coffman, "A study of interleaved memory systems," in *Proceedings of the May 5-7, 1970, Spring Joint Computer Conference on - AFIPS '70 (Spring)*. Atlantic City, New Jersey: ACM Press, 1970, p. 467.
[46] F. A. Briggs and M. Dubois, "Effectiveness of private caches in multiprocessor systems with parallel-pipelined memories," *IEEE Transactions on Computers*, vol. C-32, no. 1, pp. 48–59, Jan. 1983.
[47] B. R. Rau, "Pseudo-randomly interleaved memory," in *Proceedings of the 18th Annual International Symposium on Computer Architecture - ISCA '91*. Toronto, Ontario, Canada: ACM Press, 1991, pp. 74–83.
[48] P. Chen and D. Patterson, "Maximizing performance in a striped disk array," in *Proceedings. The 17th Annual International Symposium on Computer Architecture*. Seattle, WA, USA: IEEE Comput. Soc. Press, 1990, pp. 322–331.
[49] N. D. Enright Jerger, T. Krishna, and L.-S. Peh, *On-Chip Networks-Second Edition*, second edition ed., ser. Synthesis Lectures in Computer Architecture. Morgan & Claypool Publishers, vol. 140.
[50] W. Luo and D. Xiang, "An efficient adaptive deadlock-free routing algorithm for torus networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 800–808, Jan. 2012.
[51] M. Gorgues, D. Xiang, J. Flich, Z. Yu, and J. Duato, "Achieving balanced buffer utilization with a proper co-design of flow control and routing algorithm," in *2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*. Ferrara, Italy: IEEE, Sep. 2014, pp. 25–32.
[52] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim, "A detailed and flexible cycle-accurate network-on-chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Austin, TX, USA: IEEE, Apr. 2013, pp. 86–96.
[53] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam ; San Francisco: Morgan Kaufmann Publishers, 2004.
[54] P. Wolkotte, G. Smit, N. Kavaldjiev, J. Becker, and J. Becker, "Energy model of networks-on-chip and a bus," in *2005 International Symposium on System-on-Chip*. Tampere, Finland: IEEE, 2005, pp. 82–85.
[55] C. Meng, T. Guo, T. Xiong, Q. Chen, Z. Guo, C. Zhao, and D. Guo, "A 5-gbps serializer asic in 130 nm for high-speed front-end readout applications," *Journal of Instrumentation*, vol. 17, no. 01, p. C01072, Jan. 2022.