Heterogeneous Die-to-Die Interfaces: Enabling More Flexible Chiplet Interconnection Systems

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 architecture is one of the emerging methodologies and is believed to be scalable and economical. However, most current multi-chiplet systems are based on one uniform die-to-die interface, which severely limits flexibility. First, any interface has specific applicable workloads/scales/scenarios; therefore, chiplets with a uniform interface cannot be freely reused in different systems. Second, since modern computing systems must deal with complex and mixed tasks, the uniform interface does not cope well with flexible workloads, especially for large-scale systems.

To deal with these inflexibilities, we propose the idea of *Heterogeneous Interface (Hetero-IF)*, which allows chiplets to use two different interfaces (parallel IF and serial IF) at the same time. Hetero-IF can combine the advantages of different interfaces and cover up the disadvantages of each, thus improving flexibility and performance. However, adopting hetero-IF-based multi-chiplet interconnection systems still faces many challenges. The microarchitecture, scheduling, interconnection, and routing issues have not been discussed so far.

In this paper, we put forward two typical hetero-IF implementations: *Hetero-PHY* and *Hetero-Channel*. Based on these two implementations, detailed usages and scheduling methods are discussed. We also present the interconnection methods for hetero-IF-based multi-chiplet systems and show how to apply deadlock-free routing algorithms. Extensive evaluations, including simulation and circuit verification, are made on these systems. The experiment results show that hetero-IF provides more flexible interconnection and scheduling possibilities to achieve better performance and energy metrics under various workloads.

CCS CONCEPTS

• **Networks** → **Physical links**; **Network architectures**; • **Computer systems organization** → *Architectures*.

KEYWORDS

Network-on-Chip, Interconnection, Chiplet, Interface, Routing

ACM Reference Format:
Yinxiao Feng, Dong Xiang, and Kaisheng Ma. 2023. Heterogeneous Die-to-Die Interfaces: Enabling More Flexible Chiplet Interconnection Systems. In *56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23), October 28–November 01, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3613424.3614310

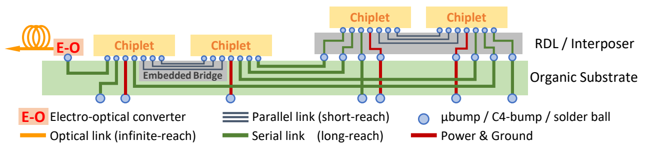

Figure 1: Multi-chiplet system: Advanced packaging technologies provide abundant interconnection possibilities.

1 INTRODUCTION

Chiplet has become one of the most popular VLSI design methodologies in recent years. With the support of advanced packaging and high-speed wireline technologies, multiple silicon dies can be integrated at high density and communication bandwidth [34, 36, 45, 48, 61]. As shown in Figure 1, since 2.5D packaging provides abundant high-quality wiring resources, chiplets can be connected together through various die-to-die interfaces[4, 11, 16, 40, 44, 47, 59, 64]. Though there are multiple interface technologies with different metrics, most current multi-chiplet systems only choose one of them according to their major requirements [31, 50, 56, 63]. Various groups in the industry are pushing for some kind of technology route as a unified standard [2, 3, 5, 7, 8, 46, 60], however, we observe two key inflexibilities caused by the uniform interface.

Motivation 1. *Chiplet reuse is limited by the uniform interface.* Many studies have shown that huge cost savings can be achieved if identical chiplets are reused among multiple systems of different scales[29, 41, 50]. However, different scenarios and scales of systems have different requirements for interfaces. Parallel interfaces such as the *Advanced Interface Bus (AIB)* are low-latency and low-power, therefore applying to high-performance systems of small scales. However, since parallel interfaces are short-reach, even with expensive advanced packages, the system can only use flat topologies such as the 2D-mesh [52, 54], which is insufficient for large-scale systems because the diameter of the network is up to $O(\sqrt{N})$ [30].

*Corresponding authors

MICRO '23, October 28–November 01, 2023, Toronto, ON, Canada
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0329-4/23/10.
https://doi.org/10.1145/3613424.3614310

Building more efficient high-radix networks [65, 66] requires long-reach high-bandwidth serial interfaces. However, the large latency and power consumption of serial interfaces are not suitable for small and energy-constrained systems. Some emerging "universal" interfaces make trade-offs between the parallel and serial interfaces, but they are not outstanding in any single metric [11, 58]. In other words, there is no such a *one-size-fits-all* universal interface for chiplet freely-reuse.

Motivation 2. *The Uniform interface is not flexible enough to handle complex and mixed network traffic.* For frequent "on-chip" communications such as the handshake, synchronization, and coherence protocols, low-latency parallel interfaces are more suitable [22, 42, 52, 68]. However, for heavy network traffic such as the all-reduce operation of large amounts of data, high-throughput and long-reach serial interfaces are better choices [37–39, 53]. In modern high-performance systems, various network traffic patterns exist simultaneously [12, 15, 17]. Choosing any kind of interface means not being able to cope well with some workloads. Motivated by the limitations of the uniform interface, we propose the ***Heterogeneous Interface*** to deal with the inflexibility in different scenarios. Nevertheless, building multi-chiplet systems through heterogeneous interfaces faces many challenges.

Challenge 1. *The design and scheduling of heterogeneous interfaces can be complex and tricky.* First, interface microarchitectures have huge impacts on systems; however, the architecture of hetero-IF-based multi-chiplet systems has not been discussed. Second, heterogeneity can lead to potential problems such as *out-of-order delivery* [55] and *heterogeneous router* [14]. Besides, the scheduling of heterogeneous interfaces is more complex (but flexible) than uniform interfaces. These issues need to be fully discussed.

Challenge 2. *Multi-chiplet systems based on heterogeneous interfaces require efficient and deadlock-free interconnection solutions.* The performance of interconnection networks is sensitive to topology and routing. However, connecting several networks-on-chiplet (NoCs) may lead to potential deadlock and congestion problems [30, 49, 69]. Heterogeneous interfaces introduce extra interconnection and routing choices, making the problems even more complicated. To fully exploit heterogeneous interfaces, methods for designing flexible interconnection schemes and applying efficient deadlock-free routing algorithms are necessary.

To address the challenges, we comprehensively discuss the implementation, microarchitecture, scheduling, interconnection, and routing issues of the heterogeneous interface and hetero-IF-based multi-chiplet systems. Moreover, we make extensive evaluations, including circuit verification, performance simulation, power estimation, and scalability evaluation. The contributions of this paper can be summarized as follows:

- For multi-chiplet systems, we analyze the limitations of the uniform die-to-die interface and propose the *heterogeneous interface* architecture. To the best of our knowledge, we are **the first** to discuss the hetero-IF-based multi-chiplet architecture.
- We present two typical heterogeneous interface implementations: *Heterogeneous PHY (Hetero-PHY)* and *Heterogeneous channel (Hetero-Channel).* The characteristics and usage of the two implementations are introduced. We also discuss the microarchitectures and overheads of the heterogeneous interface.
- We present hetero-IF-based multi-chiplet interconnection networks and give methods for applying deadlock-free routing and flexible scheduling. Evaluations show that the hetero-IF delivers huge performance and energy improvements under various workloads.

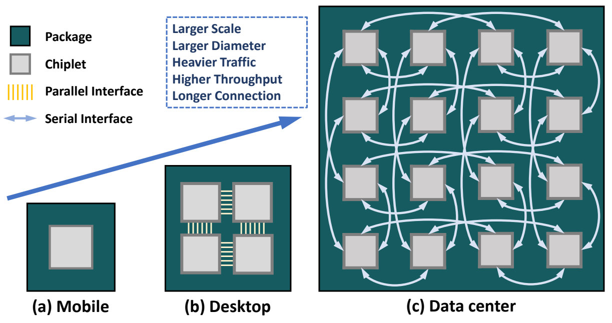

Figure 2: Chiplet reuse in systems of different scales. For different scenarios, though using the same chiplet, systems have very different integration and interconnection architectures.

2 BACKGROUND

2.1 Chiplet Architecture

The conventional VLSI system is implemented on a monolithic silicon die. However, the die area is limited by the lithographic reticle, and designing large chips is very costly[29, 62]. In recent years, packaging and high-speed wireline technologies have made great progress. Advanced packaging technologies provide a large interconnect base and a high volume of interconnection wires [34, 61]. Multiple chiplets can be interconnected and integrated at very high density and communication bandwidth. Up to now, the vast majority of current multi-chiplet systems still conservatively adopt flat topology such as 2D-mesh. Though flat topology is easy to implement, it has limited network performance and does not fully utilize the on-package interconnect resources.

Another attractive feature of chiplet architecture is *chiplet reuse.* As shown in Figure 2, identical chiplets can be used to build multiple systems of different scales. In this way, significant design costs can be saved [29, 50]. Various works try to develop such scalable systems through identical chiplets [18, 35, 43, 56, 70]. These systems use identical interconnect architectures at different scales. However, for different scenarios, systems of different scenarios and scales have different requirements for interconnection topologies and interface metrics. For example, mobile systems require low power consumption, while datacenter systems require high throughput. Feng *et al.* present a scalable method to use different topologies for different network scales [30]. However, it only adopts the uniform serial interface, which limits the latency and energy performance for small-scale systems and local communications.

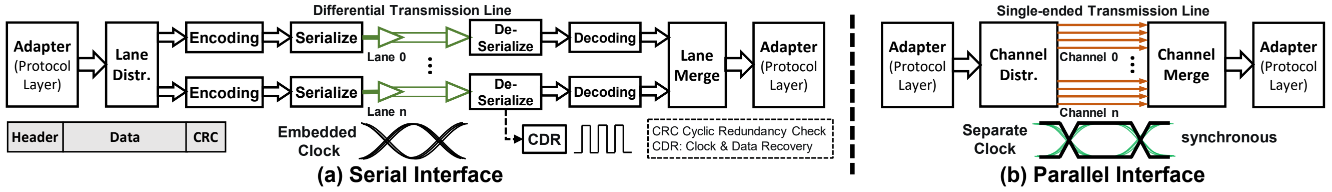

Figure 3: Microarchitecture comparison between serial interface and parallel interface.

Table 1: Specification of typical die-to-die interface

Specification	**SerDes** [2, 4, 5, 40]	**AIB** [3, 40, 60]	**BoW** [8, 10, 11, 27]	**UCIe** [7, 58, 59]
Data Rate (Gbps)	112 ✓	6.4 ✗	32	32
Latency (ns)	5.5+L_D+FEC ✗	3.5 ✓	3+L_D+FEC	2+L_D
Power (pj/bit)	2 ✗	0.5 ✓	0.7	0.3 / 1.25
Reach (mm)	50 ✓	10 ✗	50 ✓	2 / 25

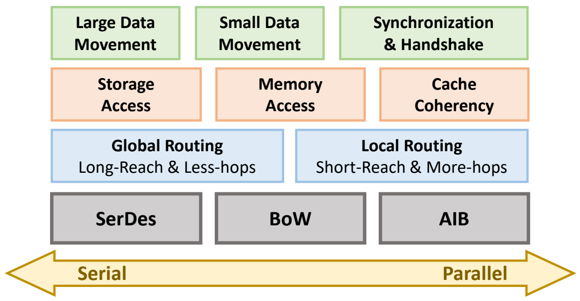

Figure 4: Different workloads have different requirements for interfaces.

2.2 Die-to-Die Interface

Since the system is partitioned into multiple chiplets, the die-to-die communication interface becomes the most significant component of the multi-chiplet system. Several technological routes have emerged in recent years [2, 3, 5, 7, 8, 46, 60]. Parameters of four typical interfaces are shown in Table 1, where L_D is digital latency. A typical interface includes several functional layers. The internal communication protocol such as AXI is first converted to external protocols such as *Compute Express Link (CXL)*. Then, the on-chip data goes through multiple processes, including encoding, serializing, scrambling, and modulating, and finally becomes analog signal leaving the chip. In this paper, we divide an interface into two layers: the physical layer (PHY) and the protocol layer (Adapter). The adapter is connected to the internal node of the chiplet, i.e., the router of the on-chip network. The PHY is led off-chip and connected to the PHY of other chiplets. chiplet-to-chiplet interfaces are classified into two categories according to the physical layer implementation.

Serial Interface. *Serializer/Deserializer (SerDes)* is the typical example of serial interface [6]. The microarchitecture of *SerDes* is shown in Figure 3(a). After encoding, the transmitter converts the on-chip parallel signal into a serial signal and sends it out at a very high frequency. After receiving the serial signal, the receiver restores it to the original parallel signal. *SerDes* uses many anti-interference technologies, such as double-terminated differential lines, CDR (clock & data recovery), and FEC (forward error correction). Therefore, the data rate and transmission distance of *SerDes* are much higher than that of the on-chip wire and off-chip none-terminated general purpose I/O. However, with so many of these anti-interference modules, *SerDes* has poor latency and energy metrics [2, 5]. *The characteristics of serial interface mainly include: high-data-rate, long-reach, high-latency, high-power.*

Parallel Interface. The typical microarchitecture of the parallel interface is shown in Figure 3(b). The implementation of the physical layer is multiple CMOS-style none-terminated I/O, and the clocks of parallel interfaces are usually separate and synchronous. *Advanced Interface Bus (AIB)* is the best-known parallel interface [3, 40, 45]. Designed specifically for chiplet, *AIB* is low-power and low-latency compared with traditional off-chip interfaces. *OpenHBI* is another parallel interface standard derived from *High Bandwidth Memory (HBM)*. With thousands of synchronized ports, *OpenHBI* has very high bandwidth, but it must be implemented on expensive silicon interposers [46]. Due to issues such as synchronization clock and signal interference, the data rate per lane and the interconnection reach of parallel interfaces are limited. *The characteristics of parallel interface mainly include: low-power, low-latency, short-reach, low-data-rate, and high-port-count (costly).*

Compromised Interface. Many emerging interfaces such as *Bunch of Wires (BoW)* [8, 10, 11, 27] and *Universal Chiplet Interconnect Express (UCIe)* [7, 58, 59] are being developed to address the shortcomings of the traditional parallel and serial interfaces. The common features of these new technologies are the improvements based on advanced packaging technologies, but they still retain the limitations of the original technology route and sacrifice some good features. For example, *BoW*, as a mixed version of the parallel and serial interfaces, has good latency and power efficiency, but the data rate is limited (32Gbps).

Interface Preference. Interfaces are crucial to multi-chiplet systems. Once the interface is determined, the shape and scope of the system are limited to some extent. For example, if the parallel interface is adopted, the topology of the system must be flat, the scale must be limited, and expensive advanced packaging technology must be used. For low-cost (normal packaging) or high-radix-network demands, new chiplets have to be re-developed. Modern applications have preferences for interfaces. As shown in Figure 4, different workloads have different requirements for interfaces. For example, the parallel interface is suitable for the frequent local

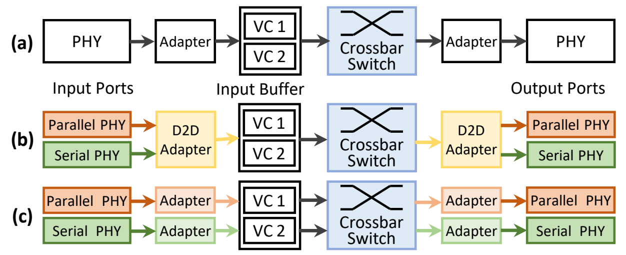

Figure 5: Heterogeneous interface architectures. (a) Uniform Interface; (b) Heterogeneous PHY; (c) Heterogeneous Channel.

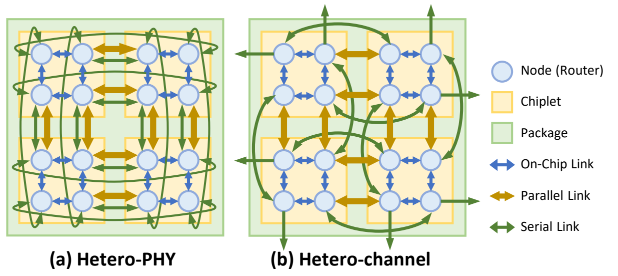

Figure 6: Interconnection networks of chiplets based on heterogeneous interfaces.

movement of small data, and the serial interface is suitable for the long-distance movement of large data. However, in most modern high-performance computing systems, these communication patterns occur simultaneously. Therefore, one uniform interface makes it hard to perfectly handle all workloads.

2.3 Routing Basics

Routing is a critical issue for the interconnection network of chiplets. Connecting several networks-on-chip (NoCs) together can lead to potential deadlocks and congestion problems[30, 49, 69]. Heterogeneous interfaces make interconnection architectures and routing algorithms more complex. Based on the existing theory, we can provide concise ideas for designing deadlock-free interconnection networks based on heterogeneous interfaces.

Lemma 1. *A connected routing function R for an interconnection network I is deadlock-free if there exists a channel subset $C_0 \subseteq C$ such that the routing subfunction $R_0(x, y) = R(x, y) \cap C_0\ \forall x, y \in N$ is connected and deadlock-free.*

Dally *et al.* and Duato *et al.* give sufficient conditions for deadlock-free adaptive routing algorithms for arbitrary interconnect networks [20, 25]. As stated in Lemma 1, if we can give a connected and deadlock-free routing function on a channel subset, then the problem of deadlock-free routing for interconnected systems is solved. The other protocol-level deadlock is mainly caused by the communication protocol [28, 51] and is not discussed further in this paper.

Livelock. Unlike deadlock, livelocked packets continue to move in the network but never reach the destinations. This is caused by the use of non-minimal paths [21, 25]. The routing algorithms and scheduling policies are supposed to be both deadlock-free and livelock-free.

3 ARCHITECTURE

As shown in Figure 5, according to whether sharing the adapter, a heterogeneous interface can be implemented in two ways: *heterogeneous PHY* and *heterogeneous channel*. Another heterogeneous interface *heterogeneous protocol* is already widely used and thus will not be discussed in this paper. For example, Compute Express Link (CXL), PCI Express, and Ethernet protocols can be used simultaneously through the SerDes PHY [2, 23, 57].

3.1 Heterogeneous PHY

As shown in Figure 5(b), heterogeneous PHY is to replace the physical layer of the uniform interface with two different PHY. The protocol layer (adapter) above the physical layer is still uniform. From the perspective of the router, the ports are still the same as traditional uniform interfaces. Traffic through the heterogeneous PHY interface is handled by the die-to-die adapter at the protocol layer.

One of the advantages of heterogeneous PHY is that it is compatible with existing interconnection architectures. Since routers do not have to be re-designed and there are few potential routing issues, the original multi-chiplet systems based on uniform interfaces can be directly migrated and are not affected by the packaging technology. The use of heterogeneous PHY is also very simple, requiring only the adapter to determine the distribution of traffic between the two heterogeneous PHYs.

Usage. The hetero-PHY interface can be used in two ways. ***1) Exclusive:*** The first way is to use only one of the interfaces depending on different scenarios. For example, only parallel interfaces are used to build low-power systems, and only serial interfaces are used to build cheaper substrate-based systems. These scenarios are usually not performance-oriented but energy-constrained or cost-constrained. Such a usage mode does not change the traditional uniform-IF-based architecture but allows the chiplet to use different interfaces in different systems, thus extending the range of applications. Though the deprecated interfaces waste some chip area, a lot of costs are saved by not re-designing the chiplet for different scenarios. ***2) Collaborative:*** The second way is to use two different PHYs simultaneously. Compared with the exclusive mode, the collaborative mode can build more flexible and efficient interconnection networks. For example, as shown in Figure 6(a), a heterogeneous 2D-torus is connected by hetero-PHY interfaces. Compared with the traditional 2D-mesh network based on uniform parallel interfaces, the hetero-PHY-based network has a smaller diameter; compared with the high radix network based on uniform serial interfaces, the Hetero-PHY-based network has lower short-reach communication latency and power consumption. Under collaborative mode, routing issues remain traditional, but a PHY-scheduling policy is necessary, which will be discussed in later sections.

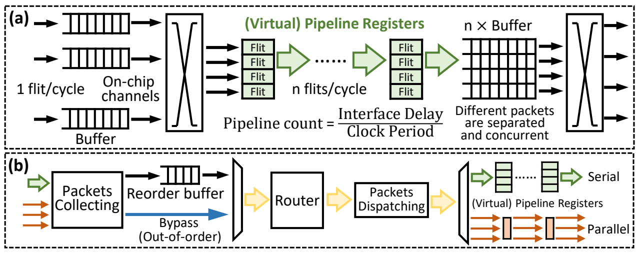
(a)
1 flit/cycle
On-chip channels
Buffer
(Virtual) Pipeline Registers
Flit
n flits/cycle
Pipeline count = $\frac{\text{Interface Delay}}{\text{Clock Period}}$
n × Buffer
Different packets are separated and concurrent
(b)
Packets Collecting
Reorder buffer
Bypass (Out-of-order)
Router
Packets Dispatching
(Virtual) Pipeline Registers
Serial
Parallel

Figure 7: Microarchitecture of the heterogeneous interface. (a) Heterogeneous router based on higher-radix crossbar; (b) Microarchitecture of the hetero-PHY adapter.

3.2 Heterogeneous Channel

As shown in Figure 5(c), the heterogeneous channel is to replace the total interface with two independent interfaces. From the perspective of the router, the original physical channel is split into two different channels. Both channels can share the original virtual channels (buffers) or have their own separate virtual channels (buffers). Traffic through the heterogeneous channel interface is handled by the router.

Compared with the hetero-PHY interface, the largest advantage of hetero-channel is the flexibility in interconnection design. Since the two channels are independent, an interface node can be interconnected with two different interface nodes, which provide numerous routing diversity and scheduling space. However, the interconnection network has to be re-designed because the router radix has changed. The router is supposed to handle routing and channel-selection issues.

Usage. Hetero-channel interface can be used just as the hetero-PHY interface. Besides, since the two different physical channels are independent, packets can either travel to adjacent chiplets with very low latency and power consumption or directly to distant chiplets via a long-reach serial interface. Physical transmission lines can be either on an advanced interposer or on a common substrate. At the same time, interconnections of multiple hierarchies are also possible. For example, as shown in Figure 6(b), four chiplets are connected into a 2D-mesh through the parallel interface. In the meantime, the serial interface connects the more distant nodes and leads out of the package for higher-hierarchy interconnection. Compared with hetero-PHY-based networks, though serial interfaces are also drawn out concurrently with parallel interfaces, they can connect to different distant nodes. Since serial interfaces are no longer restricted by short-reach parallel interfaces, hetero-channel-based multi-chiplet networks are more flexible and efficient.

4 MICROARCHITECTURE

4.1 Heterogeneous Router

Since the bandwidths of the two interfaces are usually different and the total bandwidth of the heterogeneous interface is usually larger than the on-chip links. However, traditional switching technologies are based on the homogeneous router and only allow exclusive use of the channel, thus leading to low bandwidth utilization. To make full use of the interface bandwidth, modifications to the router's microarchitecture are necessary. A few works have discussed heterogeneous router microarchitectures. They used a large buffer shared by all ports to cope with arbitrary bandwidth variances [14, 19]. However, such an architecture affects the behavior of the on-chip links and requires a complete re-design of the router. For multi-chiplet systems, only the interface ports of the network are heterogeneous. Therefore, we propose a microarchitecture that involves minor modifications to the traditional router.

As shown in Figure 7(a), a chiplet-to-chiplet interface connects two routers. The transmitter router collects packets from all on-chip ports at normal bandwidth (assumed one flit per cycle). The crossbar is higher-radix, so that multiple internal ports can concurrently send packets to the external interface. On the receiving side, we use a large multi-port input buffer to separately store the packets from different upstream channels. At the same time, the router is supposed to support concurrent routing calculations and be able to concurrently send these packets in different directions.

4.2 Hetero-PHY Adapter

Since the two interfaces of the hetero-channel are treated as two separate channels, the channel scheduling is handled by the router rather than the adapter. Therefore, we only discuss the microarchitecture design of the hetero-PHY adapter, which is shown in Figure 7(b). Similar to the superscalar pipelines [32], the hetero-PHY adapter has a front-end (transmitter) and back-end (receiver). For the transmitter side, the adapter first concurrently receives multiple packets from the router (**Fetch**). Then, necessary information, such as the type and priority, is extracted from the packet headers (**Decode**). After that, the adapter reserves physical resources according to the packet information and scheduling rules (**Dispatch**). Last, packets are sent to their respective PHYs (**Issue**).

However, the same as the superscalar pipelines, the heterogeneous interface also has *out-of-order* problems because two different physical paths have different propagation delays [55]. Before a packet is delivered through the serial interface, a later packet may have been delivered through the parallel interface, which is intolerable for some applications such as the cache coherence protocols. For messages that need to be strictly order-preserving, we adopt a reorder buffer (ROB) on the receiver side. All in-order packets are marked with additional tags and are only allowed to forward if all previous packets have been processed; otherwise, packets are kept in the reorder buffers. On the transmitter side, out-of-order packets or high-priority packets can be dispatched early through the bypass. We only allow bypass at the parallel interface because bypass introduces more delay for in-order traffic, and the parallel interface has lower latency; otherwise, it may lead to ROB overflow (unbounded waiting time). Compared with multi-path routing, the size of ROB is easy to estimate because propagation delays are deterministic [24]. We assume that the latency of the parallel path can not exceed that of the serial path (due to bypass), then the size of the ROB needs a maximum capacity of

$$S_{rob} = B_p \times (D_s - D_p), \tag{1}$$

where D_s and D_p are the delay of serial and parallel interface; B_p is the bandwidth of the parallel interface.

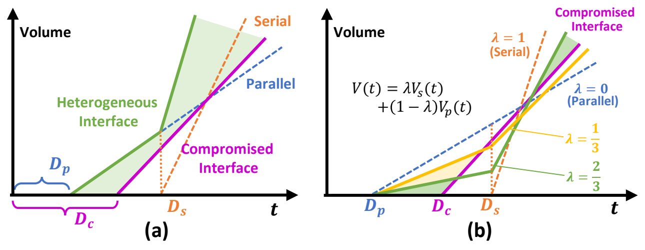

Figure 8: $V - t$ curve graph for interfaces. (a) Hetero-PHY combines the advantages of two uniform interfaces. (b) Lane/channel ratio adjustment based on requirements while controlling the total number of I/O pins.

4.3 Cost Analysis

The heterogeneous router is adopted only for the interface nodes, thus not introducing too much overhead. At the same time, hardware modifications, including multi-channel buffers, are only made on interface ports; therefore, most of the network-on-chiplet remains traditional.

The cost of the PHYs is mainly determined by the number of I/O pins. Although adopting the heterogeneous interface, the total number of I/O pins can be limited by restricting the number of lanes/channels of two PHYs. If we simply superimpose two standard PHYs, the area overhead can be large, but the total bandwidth also increases. Such an area increase is more effective than simply increasing the lane/channel number of the uniform interface. Recent work has shown limited benefit from overly increasing interface bandwidth as the bottleneck migrates to the on-chip network[30]. However, by adding a heterogeneous interface, the defects of the original uniform interface can be fundamentally solved.

Another significant overhead is the reorder buffer. By substituting some specific data to Equation 1, the capacity of the buffer is estimated to be around 10 flits, which is close to a typical packet size. The reorder buffer of the adapter can be merged with the input buffer of the router to unify the handling of other traditional out-of-order problems [55] and reduce the buffer overhead.

Although the heterogeneous interface introduces the above overheads, the increased flexibility allows chiplets to be reused in a wider range of applications, which means significant cost savings [29]. **Flexibility itself is the most significant cost saving.**

5 SCHEDULING

5.1 Bandwidth-Latency Analysis

We can use bandwidth and latency metrics to model hetero-PHY interfaces. Data volume (V) received-restored-kept in the receiver interface adapter buffer can be measured as

$$V(t) = R(B(t - D)), \tag{2}$$

where $R(x) = \max(x, 0)$, B and D are the bandwidth and total delay of the interface, respectively. $t = 0$ is the time when the transmitter adapter starts to process the data. Substituting some typical interface parameters, we can get the $V - t$ curve graph. As shown in Figure 8(a), Serial interfaces have a larger slope, and parallel interfaces have a smaller t-intercept. The compromised interface improves the shortcomings of the two uniform interfaces to some extent, but it does not fundamentally address the limitations. If we add the $V - t$ curves of the two interfaces, the resulting folds have very good properties. The hetero-PHY interface, which combines two interfaces, can transmit more data with lower latency. If we control the total number of I/O pins to be consistent, as it is the determinant of the silicon area and cost, the curves of the interfaces are shown in Figure 8(b). The hetero-PHY interface can adjust the lane/channel ratio of the two interfaces according to the requirements.

5.2 Weighted Path Length

The traditional routing algorithm is based on the minimal path of the node pairs, which is calculated by the hop number. However, for networks based on heterogeneous interfaces, the hop number only reflects limited characteristics of the path. For example, one hop through the serial interface can consume four times the latency and power of the parallel interface. One packet takes up the full bandwidth of the parallel interface, which is only a quarter of the bandwidth of the serial interface. Therefore, it is necessary to introduce more refined metrics as the routing reference. The cost of the i^{th} hop can be defined as

$$C_i = \alpha D_i + \frac{\beta}{B_i} + \gamma E_i, \tag{3}$$

where D_i is the latency, B_i is the bandwidth, E_i is the energy consumption, and α, β, γ are coefficients. Then, the "length" of a path p can be calculated as

$$L_p = \sum_{i \in p} C_i, \tag{4}$$

where i is each hop of the path. For networks based on the hetero-PHY interface, the cost of a hop is unfixed (depending on different PHY selections). Therefore, the "length" can also be expressed as $L_p = L_{p,d} + L_{p,nd}$, where $L_{p,d}$ is the deterministic length and $L_{p,nd}$ is nondeterministic length. Since it is difficult to calculate the precise length of each path, we usually count only the inter-chiplet hops, which are more costly compared with on-chip hops.

Based on the weighted length, scheduling involves three stages of routing. First, in the routing calculation stage, the routing unit is supposed to give candidate paths (output channels) based on the static properties of the network. Second, in the allocation stage, the resource allocator selects one channel based on the current dynamic properties. Third, for the hetero-PHY interface, the adapter will dispatch a specific PHY for each packet that is allocated to the current channel.

5.3 Hetero-PHY Scheduling

5.3.1 Rule-based scheduling. Rule-based scheduling is to make routing and dispatching decisions based on predefined rules. We give some specific examples of policies.

Performance-first. Network performance usually refers to the average latency of packet delivery. For the performance-first policy, parameter γ in Equation 3 is set to zero. Every hetero-PHY interface is working at full capacity: dispatch as long as there is a free PHY. Such a policy is performance-oriented but does not consider energy efficiency at all.

Energy-efficient. Different from the performance-first policy, an energy-efficient policy sets a larger weight for the energy in Equation 3. Since the parallel PHY consumes less energy than the serial PHY, the adapter always dispatches packets to the parallel PHY unless there is only the serial PHY. Such a policy provides high energy efficiency, but the network performance may be poor.

Balanced. In addition to extreme scheduling methods, there are also more balanced policies. When the adapter is dispatching packets, parallel PHY is the choice of higher priority. Under low traffic, only the parallel PHY is used. However, under heavy traffic (e.g., packets in the dispatching queue are beyond the threshold), the serial PHY is enabled to improve the network performance.

These policies provided flexibility for designing multi-chiplet systems and can be easily implemented and configured in hardware. However, rule-based scheduling does not cope well with more complex scenarios. For example, datacenter systems have mixed and unstable traffic, therefore, need more flexible scheduling methods.

5.3.2 *Application-aware scheduling.*

Application-aware scheduling is to route or dispatch packets based on packet information. Before a message is delivered by the network, it is packetized by the injection unit. Necessary information such as the destination address is encoded to the header flits. The properties of the application can actively or passively influence scheduling. Passive application-aware scheduling is to use objective characteristics of the messages. For example, the packet length and number can be used for routing and PHY-dispatching, and time-out packets can be dispatched early. Actively application-aware scheduling allows the applications to influence or control decisions. A high-priority message is noted in the packetizing stage and delivered through parallel PHY for minimal latency or through serial PHY for maximum throughput. Compared with rule-based scheduling, application-aware scheduling is more flexible for complex scenarios.

6 HETERO-CHANNEL: CASE STUDY

As we illustrate in Sec 3, compared with the hetero-PHY interface, the hetero-channel interface is more flexible. However, using hetero-channel interfaces still faces the routing challenge. Connecting several on-chip networks into a large interconnection can lead to serious deadlock and congestion problems. Extra channels provided by the hetero-channel interface introduce extra channel dependencies, which may lead to new deadlocks. Therefore, it is necessary to clarify how to apply routing algorithms on the new hardware. To better illustrate our approaches, we use a specific interconnection design as an example.

6.1 On-chip and Off-chip Interconnection

As shown in Figure 9(a), we adopt the most commonly used 2D-mesh as the on-chiplet network. All edge nodes of the 2D-mesh, also called interface nodes, are attached with external interfaces; all internal nodes, also called core nodes, do not have direct communication channels to the outside of the chiplet. The microarchitecture of the router is based on the typical virtual channel router. As shown in Figure 9(b), it contains two separate buffers (virtual channels) at each input port (physical channel). The serial interface and the parallel interface are independent and have their own virtual channel buffers. The routing of the packets for all channels can be calculated concurrently.

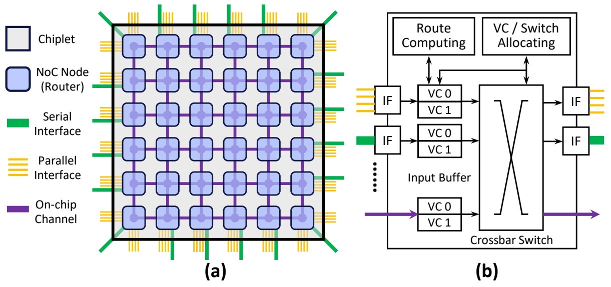

Figure 9: A specific example for hetero-channel-based chiplet. (a) On-chiplet network: 2D-mesh with interfaces all around. (b) Typical router with two virtual channels (buffers) for each channel.

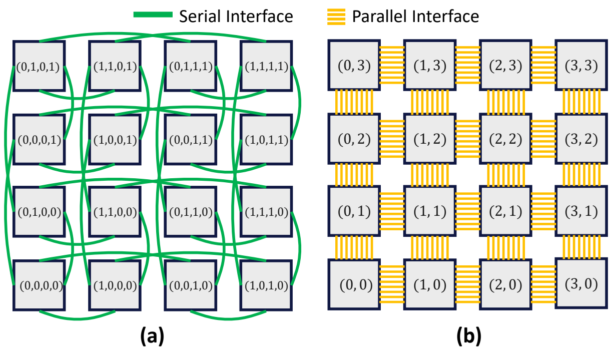

Figure 10: Two subnetworks based on different interfaces. (a) Hypercube based on the serial interface; (b) 2D-mesh based on the parallel interface.

Algorithm 1 R(x,y) BASED ON 2D-MESH SUBNETWORK

```
Input: coordinates of the current node $x$
       and the destination node $y$;
Output: candidate output channels to the next hop.
1: $C_{N,i}$ = all $i^{th}$ virtual channel of NoC channels;
2: $C_{S,j}$ = all $j^{th}$ virtual channel of serial channels;
3: $C_{P,k}$ = all $k^{th}$ virtual channel of parallel channels;
4: $C = \bigcup_{i,j,k\geq 0} (C_{N,i}, C_{S,j}, C_{P,k})$ = all (virtual) channels;
5: $C_0 = C_{N,0} \cup C_{P,0}$;
6: $R_0(x,y) = negative\_first\_routing(x,y) \subset C_0$;
7: $C_{x\to y}$ = all output channels of $x$ for all optional paths from $x$ to $y$;
8: $C_a = \left(\bigcup_{i\geq 1} C_{N,i}\right) \cup \left(\bigcup_{j\geq 0} C_{S,j}\right) \cup \left(\bigcup_{k\geq 1} C_{P,k}\right) = C - C_0$;
9: $R_a(x,y) = C_{x\to y} \cap C_a$
10: $R(x,y) = R_0(x,y) \cup R_a(x,y)$;
```

6.2 Routing Algorithm

Such chiplets can be used to build interconnection networks in an extremely flexible way. For the sake of explanation, we will discuss the parallel and serial interfaces separately. As shown in Figure 10(b), chiplets can be connected into a 2D-mesh through parallel interfaces. Due to the short-reach of the parallel interface, this is a straightforward interconnection scheme, which is commonly adopted by modern multi-chip systems [52, 56]. But 2D-mesh is considered limited for large-scale systems; thus, the long-reach serial interface compensates for the shortcomings. As shown in Figure 10(a), chiplet can be simultaneously connected into hypercube topology. The interconnection scheme draws on the method proposed by Feng *et al.* [30]. From the perspective of network diameter, the hypercube not only reduces off-chip hops but also reduces on-chip hops by cross-chiplet connections. On the other hand, from the perspective of average distance, the parallel interface effectively reduces the minimal connection latency compared with serial-only networks.

The routing algorithm for the proposed system is presented in Algorithm 1. The set of all communication virtual channels is defined as C, which consists of three parts: on-chip channels C_N, parallel interface channels C_P, and serial interface channels C_S. C_0 is a 2D-mesh subnetwork with only one virtual channel on each edge. The traditional negative-first routing is used as the deadlock-free routing function on C_0. The remaining channels, including all serial interface channels and some parallel interface and on-chip channels, can be used fully-adaptively if they belong to an optional routing path. The routing function that gives these channels is denoted as $R_a(x, y)$. The total candidate channels given by the final routing algorithm is $R_0(x, y) \cup R_a(x, y)$.

Theorem 1. *The routing Algorithm 1 is deadlock-free.*

Proof. There exists a channel subset $C_0 = C_{N,0} \cup C_{P,0}$, the routing subfunction $R' = R \cap C_0 = R_0$, is negative-first routing on 2D-mesh, which is connected and deadlock-free. Therefore, according to Lemma 1, the routing algorithm R is connected and deadlock-free. □

Since the adaptive serial channels connect nodes that are far apart, the baseline negative-first routing on 2D-mesh becomes non-minimal routing, which leads to the *livelock* problem. We solve this problem by channel switching restrictions: when a packet turns to the baseline subnetwork due to the congestion in adaptive channels of minimal paths, it is not allowed to go back to adaptive channels unless these adaptive channels belong to paths given by the baseline negative-first routing. In this way, packets always reach their destinations in limited steps.

The instanced interconnection network and routing algorithm well demonstrates how to design deadlock-free routing for networks based on hetero-channel interfaces: as long as we use some channels to build a *connected* and *deadlock-free* common network, the remaining channels can be freely scheduled. Such systems provide many good features that systems based on uniform interfaces cannot simultaneously achieve. For local communication that occurs among adjacent chiplets, parallel interfaces provide very low latency and energy consumption. For heavy global traffic, serial interfaces enable higher network throughput through cross-connectivity and high bandwidth. More importantly, the heterogeneous interface allows for more free scheduling and chiplet-reuse, thus making the system more flexible.

Table 2: Default Parameters

Parameter	**Value**
Packet Length	16 flits
Input Buffer Size	32 flits for on-chip buffers 64 flits for interface buffers
Virtual Channel Number	2 channels/link
On-chip Link Bandwidth	2 flits/cycle
Parallel Link Bandwidth	2 flits/cycle
Parallel Link Delay	5 cycles
Serial Link Bandwidth	4 flits/cycle
Serial Link Delay	20 cycles
Simulation Time	100000 cycles (10000 cycles warming up)

7 METHODOLOGY

7.1 Simulator Architecture

We use a cycle-accurate C++ simulator to evaluate the heterogeneous interface architecture. The router microarchitecture in the simulator is based on the traditional virtual-channel-based router and consists of four pipeline stages [26]: **1)** Routing; **2)** VC allocation; **3)** Switch allocation; **4)** Transmission. Under ideal conditions of zero-load, Stages 1), 2) 3) can both be completed in one clock cycle. On-chip transmission of stage 4) is completed in one cycle, but cross-chiplet transmission consumes more cycles. Cross-chiplet flow-control leads to feedback lags, and we use an additional buffer to circumvent this problem.

Interface Model. Since the signal frequency and propagation latency of the off-chip interfaces are much larger than on-chip links, it is necessary to model the interfaces as behavioral-level digital circuits of the same clock domain. As shown in Figure 7(a), we use multiple virtual pipeline registers to simulate the behavior of the off-chip interface. For each on-chip clock cycle, multiple flits move one step forward in this virtual path. The larger the bandwidth, the more concurrency is used; and the larger the latency, the more pipeline stages are inserted. The virtual pipeline stage count can be estimated at $\frac{\text{Interface Latency}}{\text{On-chip Clock Period}}$.

7.2 Simulation Setup

Parameters. The simulator's default parameters used in the experiments are shown in Table 2. The extra propagation delay of the parallel and serial interface is assumed at 5 cycles and 20 cycles. The bandwidth of the heterogeneous interface in the evaluations can be full (4-flits/cycle-serial and 2-flits/cycle-parallel) or halved (2-flits/cycle serial and 1-flits/cycle parallel). As discussed in Sec 4.3,

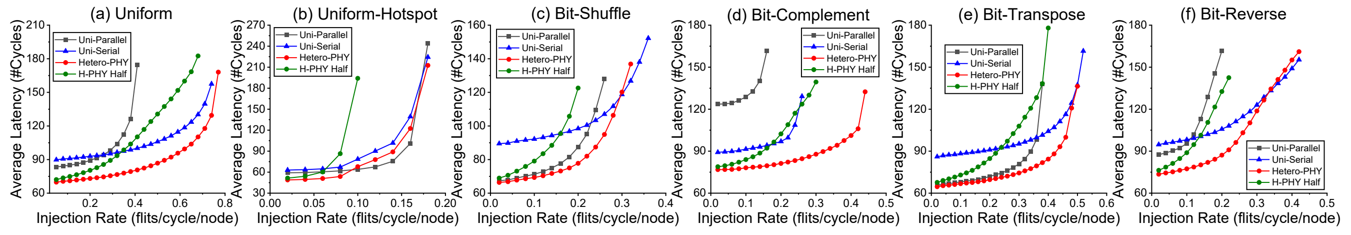

Figure 11: Performance evaluation for hetero-PHY-based interconnection network on different traffic patterns.

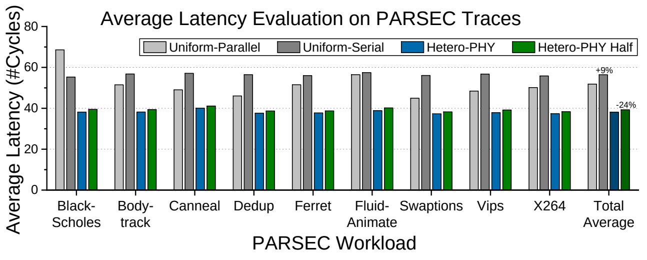

Figure 12: Performance evaluation for hetero-PHY-based interconnection network on traces of different PARSEC workloads.

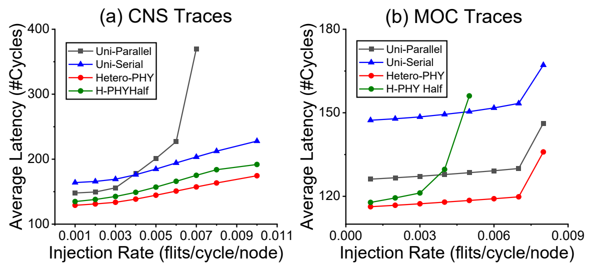

Figure 13: Performance evaluation for hetero-PHY-based interconnection network on HPC traces.

the halved hetero-IF combines two halved standard interfaces to restrict the total number of I/O pins.

Workloads. In the following evaluations, we use three main network traffic workloads. **1) Patterns.** The uniform pattern consists of random source-destination node pairs, and the uniform-hotspot traffic restricts communications only between random 10% node pairs. Other permutation traffic patterns, including bit-shuffle ($d_i = s_{(i-1) \bmod b}$), bit-complement ($d_i = \neg s_i$), bit-transpose ($d_i = s_{(i+b/2) \bmod b}$), and bit-reverse ($d_i = s_{b-i-1}$), are also evaluated [21]. **2) PARSEC traces.** The evaluated PARSEC traces are provided by Netrace, which are collected from 64-core multiprocessors running PARSEC binaries under Linux [15, 33]. During simulations, all packets are injected according to the trace time even if queuing occurs. The packet lengths are also given by the traces, which consist of two kinds of lengths: 72Bytes (9 flits) and 8Bytes (1 flit). **3) High-performance computing (HPC) traces.** The traces are collected using the open-source dumpi toolkit on Cray XE06 (Hopper) at NERSC [1, 12]. In this paper, we use two of them: one is the program for the Compressible Navier-Stokes (CNS) equations, and another is the program for the 3D method of characteristics (MOC). Both two programs use 1024 cores among all cores (153,408) of the Hopper system, and both traces have over one million packets. In our experiments, the total scale of the evaluated system is also larger than 1024 nodes.

Topology & Routing. We evaluate our hetero-IF-based interconnection methods on 2D-mesh-NoC-based chiplets shown in Figure 9(a). As shown in Figure 6(a), the hetero-PHY-based chiplets can be connected into 2D-mesh through parallel interfaces and simultaneously into 2D-torus through serial interfaces. As shown in Figure 10, the hetero-channel-based chiplets can be connected into 2D-mesh through parallel interfaces and simultaneously into hypercube through serial interfaces. **The baseline systems are uniform-parallel-IF-based 2D-mesh as well as uniform-serial-IF-based 2D-torus (compared with hetero-PHY) and hypercube (compared with hetero-channel).** Negative-first-based adaptive routing is adopted for 2D-mesh and 2D-torus, and minus-first-based adaptive routing is adopted for hypercube (reproduced from [30]).

7.3 Circuit Implementation

We also verify the heterogeneous interface by actual implementations. Post-synthesis analysis is made at TSMC-12nm. **1) Hetero-PHY adapter RX (reorder buffer).** We use a FIFO to buffer the flits (data and sequence number (SN)) from the parallel-PHY and wait for flits with earlier SN to arrive from the serial-PHY. **2) Hetero-PHY adapter TX (multi-width FIFO).** We implement a FIFO that can read/write multiple flits in one cycle. The control logic decides how many flits to read based on the current state. Specifically, we implement the *balance scheduling*, as referred in Sec. 5.3.1: if data in the FIFO reaches half of the capacity, read three flits, one to the parallel-PHY and two to the serial-IF; otherwise, read one flit to the parallel-PHY. **3) Heterogeneous router (high-radix router).** We let the parallel-IF use the original port and added two extra ports, including routing computing logic, for the serial-IF. The router architecture is the canonical VC router [9, 13, 14, 21].

8 EVALUATION

8.1 Performance Evaluation

8.1.1 Hetero-PHY-based Network. We adopt the rule-based *balanced* scheduling policy for hetero-PHY-based 2D-torus in the evaluations, i.e., parallel PHYs are used at higher priority.

Traffic patterns. First, we evaluate six common traffic patterns on a medium-scale system (4×4 chiplets, 4×4 nodes per chiplet, 256

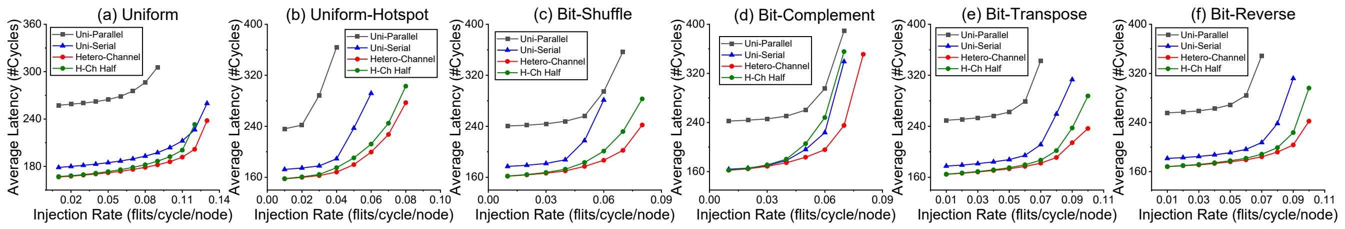

Figure 14: Performance evaluation for hetero-channel-based interconnection network on different traffic patterns.

nodes in total). As shown in Figure 11, under light traffic, uniform-serial-IF-based 2D-torus has poor latency performance because the delay of the serial interface is high. For uniform-parallel-IF-based 2D-mesh, as the diameter is long and the bisection bandwidth is limited, the throughput performance is poor. Our hetero-PHY-based 2D-torus with full bandwidth has better latency and saturation injection rate at all traffic patterns. However, the hetero-PHY-based 2D-torus with halved bandwidth has poor latency performance under heavy traffic because the wraparound links are halved-bandwidth serial-only interfaces, which severely affects the performance.

PARSEC traces. Besides traffic patterns, real-world workloads are also evaluated. As the PARSEC traces are collected from 64-core multiprocessors, we adopt the same scale system (4×4 chiplets, 2×2 nodes per chiplet, 64 nodes in total). As shown in Figure 12, under all workloads, the hetero-PHY-based 2D-torus has better latency performance. The full-bandwidth system and half-bandwidth system have similar results because wraparound packets do not occupy a large proportion of PARSEC traces. The uniform-parallel-IF-based 2D-mesh has better performance than the uniform-serial-IF-based 2D-torus because the delay of the serial interface (20 cycles) is dominant for small-scale networks. The latency variance of hetero-IF-based networks is lower than in the case of uniform-IF-based networks, indicating that less congestion occurs during the simulations.

HPC traces. We also evaluate our interconnection method on a large-scale system (6×6 chiplets, 6×6 nodes per chiplet, 1296 nodes in total). As shown in Figure 13, for the CNS program, the hetero-PHY-based 2D-torus has better throughput than the uniform-parallel-IF-based 2D-mesh network and has better latency than the uniform-serial-IF-based 2D-torus network. For the MOC program, the hetero-PHY-based 2D-torus also has a latency advantage, but the saturation injection rates of the three networks are the same. The half-bandwidth system has half the saturation injection rate, indicating the interface is fully used for the MOC program.

Summary. From the results, we can see that hetero-PHY performs better than traditional uniform interfaces under various workloads. However, since the parallel and serial interfaces are bound together, there is still inflexibility in practice. Halving the interface bandwidth (restricting the total number of I/O pins) in many scenarios can affect performance.

8.1.2 Hetero-Channel-based Network. We adopt the routing algorithm proposed in Sec 6.2 for the hetero-channel-based interconnection network. As discussed earlier in Sec 3.2, hetero-channel is used for large-scale systems, such as the wafer-level datacenter system. Therefore, we build a large-scale system consisting of 3136 nodes (7×7 nodes per chiplet, 8×8[2^6] chiplets). Rule-based *balanced* scheduling is used based on cross-chiplet hops from the source chiplet to the destination chiplet. The channel selection function can be described as

$$SS = \begin{cases} \text{Serial-IF-based cube}, & \#H_P - \#H_S > 0 \\ \text{Parallel-IF-based mesh}, & \#H_P - \#H_S \leq 0 \end{cases} \quad (5)$$

where SS is the selected subnetwork, $\#H_P$ is the parallel link hop count in the 2D-mesh subnetwork, and $\#H_S$ is the serial link hop count in the hypercube subnetwork. This function results in the lowest total number of cross-chiplet hops.

Traffic patterns. We evaluate the six traffic patterns on the systems. As shown in Figure 14, under various traffic patterns, the uniform-serial-IF-based hypercube network has better performance than the uniform-parallel-IF-based 2D-mesh network. Our hetero-channel-based network with a parallel-IF-based subnetwork has even better performance than the serial-IF-only network. That's because when a message is approaching the destination, it can turn to use the low-latency parallel interface rather than always using the long-reach serial interface. Since high-radix networks have lower requirements on link bandwidth, the half-bandwidth interfaces do not affect performance too much.

HPC traces. In the evaluation of the HPC traces, we use the core nodes of each chiplet. As shown in Figure 15, for the CNS program, the hetero-channel-based network has better throughput and latency, which is similar to the result of the hetero-PHY evaluation. For the MOC program, the hetero-channel-based network and uniform-parallel-IF-based 2D-mesh have similar throughput results. There is a small twist at the beginning because the parallel interface is queued up, and more serial interfaces are used. The half-bandwidth interfaces also do not affect the performance.

Summary. From the results, we can see that hetero-channel also performs better than traditional uniform interfaces under various

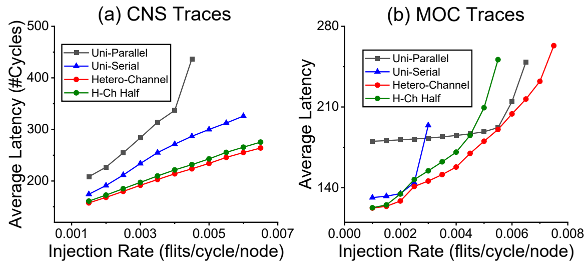

Figure 15: Performance evaluation for hetero-channel-based interconnection network on HPC traces.

workloads. Compared with the hetero-PHY interface, the hetero-channel interface allows for completely different high-radix topologies, thus reducing the bandwidth requirements. Therefore, the hetero-channel interface enables more flexibility in building multi-chiplet interconnect systems while reducing the interface overhead.

Table 3: Average latency reduction of hetero-IF compared with uniform-parallel-IF / uniform-serial-IF.

Scale (On-Chip)	Hetero-PHY	Hetero-Channel
4×(2×2)	17.3% / 21.7%	/
16×(2×2)	17.5% / 30.0%	/
16×(4×4)	16.4% / 21.8%	9.6% / 22.2%
16×(6×6)	19.3% / 17.9%	15.5% / 19.8%
64×(7×7)	35.8% / 20.5%	46.4% / 13.1%

8.1.3 Scalability Evaluation. We do further evaluations on the scalability. Uniform traffic at 0.1 flits/cycle/node is evaluated on 5 systems of different on-chip and off-chip scales. We count the average latency reductions of hetero-IF-based networks compared with uniform-parallel-IF-based and uniform-serial-IF-based networks.

As shown in Table 3, heterogeneous interfaces achieve 9.6% - 46.4% lower latency for systems of different scales. The results show that the heterogeneous interface has good scalability.

8.2 Post-Synthesis Analysis

The RX adapter includes a 64bit-width 16-depth FIFO and the counting logic; the TX adapter includes a same-size FIFO (queue) with 3 concurrent read/write ports and the scheduling logic. Compared with the regular adapter, the reordering logic adds one extra cycle. As shown in Table 4, the area/energy overhead of the adapter is small, and the module can run at a high frequency (1.85GHz). The overhead of the heterogeneous router is relatively high. Adding extra 2 concurrent ports increases area by 45% and power by 33%. Nevertheless, the router's operating frequency was not significantly affected, and power/area is still proportional to the throughput.

Table 4: Post-Synthesis Analysis Result

Module		Area (um2)	Power	Frequency Critical Path
Adapter	RX	1389	1.14mW 3.2fJ/bit	up to 1.85GHz 0.36ns
	TX	1849	0.78mW 3.3fJ/bit	up to 1.85GHz 0.37ns
Router	Regular	7007	2.19mW	up to 1.20GHz 0.65ns
	Hetero-geneous	10155	2.92mW	up to 1.16GHz 0.67ns

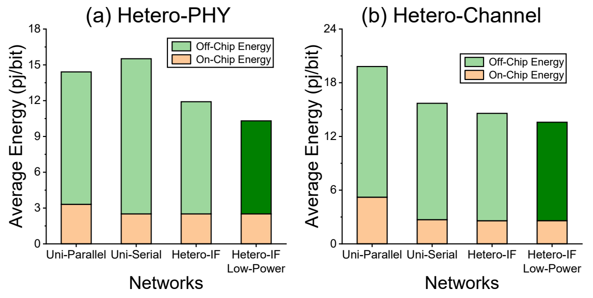

Figure 16: Average energy consumption on the uniform traffic.

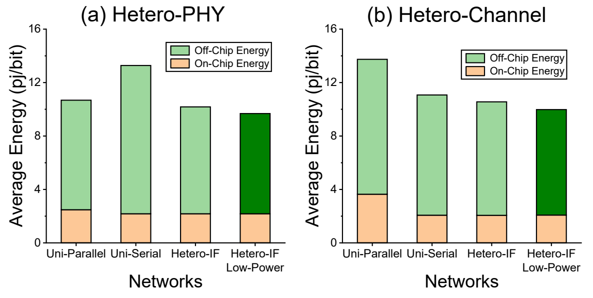

Figure 17: Average energy consumption on the HPC traffic.

8.3 Energy Evaluation

We also evaluate the energy consumption of the hetero-IF-based interconnection networks. The overhead of the off-chip interfaces is estimated as 1 pj/bit for the parallel interface and 2.4 pj/bit for the serial interface. We count the energy consumed by each packet on the path and calculate the average.

Uniform traffic. The evaluated traffic injection rate is 0.1. The hetero-PHY-based system is the large-scale 2D system mentioned in Sec 8.1.1. As shown in Figure 16(a), the uniform-parallel-IF-based 2D-mesh has poor on-chip energy performance because it has more average hops compared with 2D-torus. The energy performance of the uniform-serial-IF-based 2D-torus is suffered from the high-energy interface. The hetero-PHY-based torus achieves fewer hop counts and lower hop cost simultaneously, thus achieving lower energy consumption. If we restrict the scheduling method to *energy-efficient*, which is described in Sec 5.3.1, 7% further energy reduction is achieved. As shown in Figure 16(b), The hetero-channel-based interconnection system has the same topology and scale as in Sec 8.1.2. The hetero-channel interface with *energy-efficient* scheduling achieves 31% and 13% lower energy compared with uniform-parallel-IF and uniform-serial-IF.

HPC workloads. We also evaluate the energy performance under real-world traffic (MOC traces). The topology and scale are the same as in Sec 8.1. As shown in Figure 17(a), the hetero-PHY-based interconnection network achieves 9% lower power consumption compared with the uniform-parallel-IF-based 2D-mesh network. As shown in Figure 17(b), the hetero-channel-IF with *energy-efficient* scheduling achieves 27% and 10% lower energy compared with uniform-parallel-IF and uniform-serial-IF.

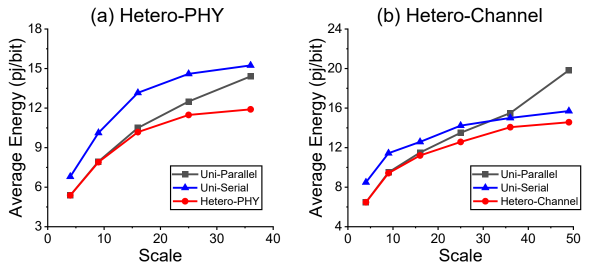

Figure 18: Average energy consumption for different traffic scales

Flexibility for different traffic scales. In the above experiments, the energy performance of hetero-IF-based large-scale systems is similar to the uniform-serial-IF-based systems. However, for modern high-performance computing systems, there are massive local communications, which are not suitable for using serial interfaces. We evaluated the energy performance of local communication in large-scale systems by limiting the position and scale of the communication nodes. As shown in Figure 18, the horizontal coordinates indicate different local communication scales, and the evaluated uniform traffic injection rate is 0.01 flits/cycle/node. For short-reach local traffic, the energy performance of uniform-serial-IF-based systems is not as good as for full-scale traffic. Hetero-IF-based systems, with both low-power, short-reach connectivity and cross-chip, long-reach connectivity, provide better energy performance at all scales of workloads.

9 ANALYSIS

Why hetero-IF is effective? Extensive evaluation results show that heterogeneous interfaces bring good performance in both latency and energy. The most essential reason for this improvement is that the hetero-IF combines the advantages of two different interfaces and covers up the disadvantages of both. More specifically, the hetero-IF provides channel adaptivity and diversity, which allows packets to traverse paths with fewer hops, lower latency, less energy, and less congestion.

What are the applicable scenarios? Weighing the costs and benefits, we believe hetero-IF is applicable in the following scenarios: 1) Large-scale high-performance computing systems with complex and mixed network traffic. 2) Chiplets which are reused among various systems with different scales and applications. 3) Multi-chiplet systems which are supposed to be highly customized and allow software-defined scheduling. However, hetero-IF is not applicable in: 1) Scenario where the energy and area are extremely limited. 2) Dedicated systems with fixed and simplex communication modes.

Fault tolerance. Conventional fault-tolerant routing algorithms can also be applied to hetero-IF-based interconnection networks [65, 67]. Since hetero-IF provides more channel diversity and adaptivity, it may improve the system's fault tolerance.

10 SUMMARY

The same as all architectures, the chiplet architecture also seeks generality. People have always expected one standard chiplet-to-chiplet interface to cover all scenarios. However, it has been proven in practice that the uniform interface can significantly limit the chiplet architecture. Therefore, we propose and discuss a new architecture: *Heterogeneous Interface*, which makes multi-chiplet systems more flexible. In general, flexibility is reflected in three aspects:

- *Flexibility in* ***interconnection***. First, since the connection is no longer limited by the physical properties of the interface, the topology of multi-chiplet networks can be more flexible. Second, there is richer path diversity for hetero-IF-based networks, which improves the network performance under complex traffic.
- *Flexibility in* ***scheduling***. Heterogeneous interfaces allow different packets to go through different physical channels at different times, which greatly enhances the flexibility of system scheduling. By rule-based or application-aware scheduling policies, better metrics are achieved.
- *Flexibility in* ***economy***. First, the choice of packaging options can be more flexible, meaning that the appropriate package can be selected based on cost constraints. Second, the application scope of chiplets has been expanded, meaning that more systems can be composed with fewer chiplets, which has proven to be economical.

In this paper, two typical implementations of heterogeneous interfaces and their usages are discussed. The overheads are analyzed, and a few practical scheduling methods are introduced. Also, interconnection and routing methods for hetero-IF-based networks are presented and discussed. Extensive evaluations demonstrate that hetero-IF does deliver significant performance improvements and flexibility gains.

11 ACKNOWLEDGE

This research is partially supported by National Key R&D Program of China (2022YFB2804103), Key Research and Development Program of Shaanxi (2021ZDLGY01-05), National Natural Science Foundation of China (20211710187, 31970972), Tsinghua University Dushi Program, Tsinghua University Talent Program, and research projects from Ant Group and Lenovo.

REFERENCES

[1] 2016. Characterization of the DOE Mini-Apps. https://portal.nersc.gov/project/CAL/overview.htm.
[2] 2017. Serial Interface for Data Converters (JEDEC204C). https://www.jedec.org/standards-documents/docs/jesd-204a.
[3] 2020. AIB Specification 2.0. https://github.com/chipsalliance/AIB-specification/blob/master/AIB_Specification%202_0.pdf.
[4] 2021. DesignWare Die-to-Die 112G USR/XSR PHY & Die-to-Die Controller.
[5] 2022. Common Electrical I/O (CEI) - Electrical and Jitter Interoperability Agreements for 6G+ Bps, 11G+ Bps, 25G+ Bps, 56G+ Bps and 112G+ Bps I/O. https://www.oiforum.com/wp-content/uploads/OIF-CEI-5.0.pdf.
[6] 2022. SerDes. https://en.wikipedia.org/wiki/SerDes.
[7] 2022. Universal Chiplet Interconnect Express (UCIe) Specification Revision 1.0. https://www.uciexpress.org/specification.
[8] 2023. Bunch of Wires PHY Specification. https://opencomputeproject.github.io/ODSA-BoW/bow_specification.html.
[9] anan-cn. 2023. Anan-Cn/Open-Source-Network-on-Chip-Router-RTL. https://github.com/anan-cn/Open-Source-Network-on-Chip-Router-RTL.

[10] Shahab Ardalan, Halil Cirit, Ramin Farjad, Mark Kuemerle, Ken Poulton, Suresh Subramanian, and Bapiraju Vinnakota. 2020. Bunch of Wires: An Open Die-to-Die Interface. In *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, Piscataway, NJ, USA, 9–16. https://doi.org/10.1109/HOTI51249.2020.00017

[11] Shahab Ardalan, Ramin Farjadrad, Mark Kuemerle, Ken Poulton, Suresh Subramaniam, and Bapiraju Vinnakota. 2021. An Open Inter-Chiplet Communication Link: Bunch of Wires (BoW). *IEEE Micro* 41, 1 (Jan. 2021), 54–60. https://doi.org/10.1109/MM.2020.3040410

[12] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the Complexity of Traffic Traces and Implications. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 1 (May 2020), 1–29. https://doi.org/10.1145/3379486

[13] Daniel U. Becker. [n. d.]. Efficient Microarchitecture for Network-on-Chip Routers.

[14] Yaniv Ben-Itzhak, Israel Cidon, Avinoam Kolodny, Michael Shabun, and Nir Shmuel. 2015. Heterogeneous NoC Router Architecture. *IEEE Transactions on Parallel and Distributed Systems* 26, 9 (Sept. 2015), 2479–2492. https://doi.org/10.1109/TPDS.2014.2351816

[15] Christian Bienia. 2011. *Benchmarking Modern Multiprocessors.* Ph. D. Dissertation. Princeton University, USA.

[16] Anthony Chan Carusone, Behzad Dehlaghi, Rudy Beerkens, and Davide Tonietto. 2016. Ultra-Short-Reach Interconnects for Package-Level Integration. In *2016 IEEE Optical Interconnects Conference (OI)*. IEEE, San Diego, CA, USA, 10–11. https://doi.org/10.1109/OIC.2016.7483009

[17] Siva Shankar Chandrasekaran. 2017. *Understanding Traffic Characteristics in a Server to Server Data Center Network.* Ph. D. Dissertation. Rochester Institute of Technology.

[18] Bill Chang, Rajiv Kurian, Doug Williams, and Eric Quinnell. 2022. DOJO: Super-Compute System Scaling for ML Training. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. IEEE, Cupertino, CA, USA, 1–45. https://doi.org/10.1109/HCS55958.2022.9895625

[19] H. Jonathan Chao and Bin Liu. 2007. *High Performance Switches and Routers.* Wiley-Interscience, Hoboken, N.J.

[20] William J. Dally and Charles L. Seitz. 1987. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Trans. Comput.* C-36, 5 (May 1987), 547–553. https://doi.org/10.1109/TC.1987.1676939

[21] William J. Dally and Brian Towles. 2004. *Principles and Practices of Interconnection Networks.* Morgan Kaufmann Publishers, Amsterdam ; San Francisco.

[22] Jonas Diemer, Rolf Ernst, and Michael Kauschke. 2010. Efficient Throughput-Guarantees for Latency-Sensitive Networks-on-Chip. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, Taipei, Taiwan, 529–534. https://doi.org/10.1109/ASPDAC.2010.5419828

[23] Kevin Drucker, Dharmesh Jani, Ishwar Agarwal, Gary Miller, Millind Mittal, Robert Wang, and Bapiraju Vinnakota. 2020. The Open Domain-Specific Architecture. In *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, Piscataway, NJ, USA, 25–32. https://doi.org/10.1109/HOTI51249.2020.00019

[24] Gaoming Du, Miao Li, Zhonghai Lu, Minglun Gao, and Chunhua Wang. 2014. An Analytical Model for Worst-Case Reorder Buffer Size of Multi-Path Minimal Routing NoCs. In *2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*. IEEE, Ferrara, Italy, 49–56. https://doi.org/10.1109/NOCS.2014.7008761

[25] José Duato, Sudhakar Yalamanchili, and Lionel M. Ni. 2003. *Interconnection Networks: An Engineering Approach* (rev. printing ed.). Morgan Kaufmann, San Francisco, CA.

[26] Natalie D. Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. 2018. *On-Chip Networks-Second Edition* (second edition ed.). Synthesis Lectures in Computer Architecture, Vol. 140. Morgan & Claypool Publishers, San Rafael, California.

[27] Ramin Farjadrad, Mark Kuemerle, and Bapi Vinnakota. 2020. A Bunch-of-Wires (BoW) Interface for Interchiplet Communication. *IEEE Micro* 40, 1 (Jan. 2020), 15–24. https://doi.org/10.1109/MM.2019.2950352

[28] Hossein Farrokhbakht, Henry Kao, Kamran Hasan, Paul V. Gratz, Tushar Krishna, Joshua San Miguel, and Natalie Enright Jerger. 2021. Pitstop: Enabling a Virtual Network Free Network-on-Chip. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Seoul, Korea (South), 682–695. https://doi.org/10.1109/HPCA51647.2021.00063

[29] Yinxiao Feng and Kaisheng Ma. 2022. Chiplet Actuary: A Quantitative Cost Model and Multi-Chiplet Architecture Exploration. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. ACM, San Francisco California, 121–126. https://doi.org/10.1145/3489517.3530428

[30] Yinxiao Feng, Dong Xiang, and Kaisheng Ma. 2023. A Scalable Methodology for Designing Efficient Interconnection Network of Chiplets. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Montreal, QC, Canada, 1059–1071. https://doi.org/10.1109/HPCA56546.2023.10070981

[31] Wilfred Gomes, Altug Koker, Pat Stover, Doug Ingerly, Scott Siers, Srikrishnan Venkataraman, Chris Pelto, Tejas Shah, Amreesh Rao, Frank O'Mahony, Eric Karl, Lance Cheney, Iqbal Rajwani, Hemant Jain, Ryan Cortez, Arun Chandrasekhar, Basavaraj Kanthi, and Raja Koduri. 2022. Ponte Vecchio: A Multi-Tile 3D Stacked Processor for Exascale Computing. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. IEEE, San Francisco, CA, USA, 42–44. https://doi.org/10.1109/ISSCC42614.2022.9731673

[32] Antonio González, Fernando Latorre, and Grigorios Magklis. 2010. *Processor Microarchitecture: An Implementation Perspective.* Synthesis Lectures on Computer Architecture, Vol. 5.

[33] Joel Hestness, Boris Grot, and Stephen W. Keckler. 2010. Netrace: Dependency-Driven Trace-Based Network-on-Chip Simulation. In *Proceedings of the Third International Workshop on Network on Chip Architectures*. ACM, Atlanta Georgia USA, 31–36. https://doi.org/10.1145/1921249.1921258

[34] P. K. Huang, C. Y. Lu, W. H. Wei, Christine Chiu, K. C. Ting, Clark Hu, C.H. Tsai, S. Y. Hou, W. C. Chiou, C. T. Wang, and Douglas Yu. 2021. 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)*. IEEE, San Diego, CA, USA, 101–104. https://doi.org/10.1109/ECTC32696.2021.00028

[35] Drago Ignjatovic, Daniel W. Bailey, and Ljubisa Bajic. 2022. The Wormhole AI Training Processor. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. IEEE, San Francisco, CA, USA, 356–358. https://doi.org/10.1109/ISSCC42614.2022.9731633

[36] D. B. Ingerly, K. Enamul, W. Gomes, D. Jones, K. C. Kolluru, A. Kandas, G.-S. Kim, H. Ma, D. Pantuso, C.F. Petersburg, M. Phen-givoni, S. Amin, A. M. Pillai, A. Sairam, P. Shekhar, P. Sinha, P. Stover, A. Telang, Z. Zell, L. Aryasomayajula, A. Balankutty, D. Borst, A. Chandra, K. Cheemalapati, C. S. Cook, and R. Criss. 2019. Foveros: 3D Integration and the Use of Face-to-Face Chip Stacking for Logic Devices. In *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, San Francisco, CA, USA, 19.6.1–19.6.4. https://doi.org/10.1109/IEDM19573.2019.8993637

[37] Norman P Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. In *2023 ACM/IEEE 50th Annual International Symposium on Computer Architecture (ISCA)*.

[38] Norman P Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, et al. 2021. Ten lessons from three generations shaped google's tpuv4i: Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 1–14.

[39] Divya Kiran Kadiyala, Saeed Rashidi, Taekyung Heo, Abhimanyu Rajeshkumar Bambhaniya, Tushar Krishna, and Alexandros Daglis. 2022. COMET: A Comprehensive Cluster Design Methodology for Distributed Deep Learning Training. arXiv:2211.16648 [cs.DC]

[40] David Kehlet. [n. d.]. Accelerating Innovation Through a Standard Chiplet Interface: The Advanced Interface Bus (AIB). https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/accelerating-innovation-through-aib-whitepaper.pdf.

[41] Jinwoo Kim, Gauthaman Murali, Heechun Park, Eric Qin, Hyoukjun Kwon, Venkata Chaitanya Krishna Chekuri, Nael Mizanur Rahman, Nihar Dasari, Arvind Singh, Minah Lee, Hakki Mert Torun, Kallol Roy, Madhavan Swaminathan, Saibal Mukhopadhyay, Tushar Krishna, and Sung Kyu Lim. 2020. Architecture, Chip, and Package Codesign Flow for Interposer-Based 2.5-D Chiplet Integration Enabling Heterogeneous IP Reuse. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 11 (Jan. 2020), 2424–2437. https://doi.org/10.1109/TVLSI.2020.3015494

[42] Yuan Li and Ahmed Louri. 2021. ALPHA: A Learning-Enabled High-Performance Network-on-Chip Router Design for Heterogeneous Manycore Architectures. *IEEE Transactions on Sustainable Computing* 6, 2 (April 2021), 274–288. https://doi.org/10.1109/TSUSC.2020.2981340

[43] Sean Lie. 2021. Multi-Million Core, Multi-Wafer AI Cluster. In *2021 IEEE Hot Chips 33 Symposium (HCS)*. IEEE, Palo Alto, CA, USA, 1–41. https://doi.org/10.1109/HCS52781.2021.9567153

[44] Joe Lin, C. Key Chung, C. F. Lin, Ally Liao, Ying Ju Lu, Jia Shuang Chen, and Daniel Ng. 2020. Scalable Chiplet Package Using Fan-Out Embedded Bridge. In *2020 IEEE 70th Electronic Components and Technology Conference (ECTC)*. IEEE, Orlando, FL, USA, 14–18. https://doi.org/10.1109/ECTC32862.2020.00015

[45] Chester Liu, Jacob Botimer, and Zhengya Zhang. 2021. A 256Gb/s/Mm-Shoreline AIB-Compatible 16nm FinFET CMOS Chiplet for 2.5D Integration with Stratix 10 FPGA on EMIB and Tiling on Silicon Interposer. In *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, Austin, TX, USA, 1–2. https://doi.org/10.1109/CICC51472.2021.9431555

[46] Kenneth Ma, Andy He, Tom Wilson, Moo Sung Chae, Jeffrey Bostak, Peter Nyasulu, Brian Worobey, Anh Nguyen, Millind Mittal, and Xiaobao Wang. 2021. OpenHBI Specification Version 1.0. https://www.opencompute.org/documents/odsa-openhbi-v1-0-spec-rc-final-1-pdf.

[47] Xiaohan Ma, Ying Wang, Yujie Wang, Xuyi Cai, and Yinhe Han. 2022. Survey on Chiplets: Interface, Interconnect and Integration Methodology. *CCF Transactions on High Performance Computing* 4, 1 (March 2022), 43–52. https://doi.org/10.1007/s42514-022-00093-0

[48] Ravi Mahajan, Robert Sankman, Kemal Aygun, Zhiguo Qian, Ashish Dhall, Jonathan Rosch, Debendra Mallik, and Islam Salama. 2019. Embedded Multi-Die Interconnect Bridge (EMIB): A Localized, High Density, High Bandwidth Packaging Interconnect. In *Advances in Embedded and Fan-Out Wafer-Level Packaging Technologies*, Beth Keser and Steffen Kroehnert (Eds.). John Wiley & Sons, Inc., Hoboken, NJ, USA, 487–499. https://doi.org/10.1002/9781119313991.ch23

[49] Pritam Majumder, Sungkeun Kim, Jiayi Huang, Ki Hwan Yum, and Eun Jung Kim. 2021. Remote Control: A Simple Deadlock Avoidance Scheme for Modular Systems-on-Chip. *IEEE Trans. Comput.* 70, 11 (Nov. 2021), 1928–1941. https://doi.org/10.1109/TC.2020.3029682

[50] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. 2021. 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)*. IEEE, Valencia, Spain, 57–70. https://doi.org/10.1109/ISCA52012.2021.00014

[51] Vijay Nagarajan, Daniel J. Sorin, Mark D. Hill, and David A. Wood. 2020. *A Primer on Memory Consistency and Cache Coherence, Second Edition.* Synthesis Lectures on Computer Architecture, Vol. 15.

[52] Nevine Nassif, Ashley O. Munch, Carleton L. Molnar, Gerald Pasdast, Sitaraman V. Lyer, Zibing Yang, Oscar Mendoza, Mark Huddart, Srikrishnan Venkataraman, Sireesha Kandula, Rafi Marom, Alexandra M. Kern, Bill Bowhill, David R. Mulvihill, Srikanth Nimmagadda, Varma Kalidindi, Jonathan Krause, Mohammad M. Haq, Roopali Sharma, and Kevin Duda. 2022. Sapphire Rapids: The Next-Generation Intel Xeon Scalable Processor. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, San Francisco, CA, USA, 44–46. https://doi.org/10.1109/ISSCC42614.2022.9731107

[53] Akira Nukada. 2021. Performance Optimization of Allreduce Operation for Multi-GPU Systems. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, Orlando, FL, USA, 3107–3112. https://doi.org/10.1109/BigData52589.2021.9672073

[54] Saptadeep Pal, Jingyang Liu, Irina Alam, Nicholas Cebry, Haris Suhail, Shi Bu, Subramanian S. Iyer, Sudhakar Pamarti, Rakesh Kumar, and Puneet Gupta. 2021. Designing a 2048-Chiplet, 14336-Core Waferscale Processor. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, San Francisco, CA, USA, 1183–1188. https://doi.org/10.1109/DAC18074.2021.9586194

[55] Maurizio Palesi, Rickard Holsmark, Xiaohang Wang, Shashi Kumar, Mei Yang, Yingtao Jiang, and Vincenzo Catania. 2010. An Efficient Technique for In-Order Packet Delivery with Adaptive Routing Algorithms in Networks on Chip. In *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*. IEEE, Lille, France, 37–44. https://doi.org/10.1109/DSD.2010.53

[56] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2019. Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, Columbus OH USA, 14–27. https://doi.org/10.1145/3352460.3358302

[57] Debendra Das Sharma. 2022. Compute Express Link®: An Open Industry-Standard Interconnect Enabling Heterogeneous Data-Centric Computing. In *2022 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, CA, USA, 5–12. https://doi.org/10.1109/HOTI55740.2022.00017

[58] Debendra Das Sharma. 2022. UCIe White Paper. https://www.uciexpress.org/general-8.

[59] Debendra Das Sharma, Gerald Pasdast, Zhiguo Qian, and Kemal Aygun. 2022. Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations With Chiplets at Package Level. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 12, 9 (Sept. 2022), 1423–1431. https://doi.org/10.1109/TCPMT.2022.3207195

[60] Farhana Sheikh and David Kehlet. 2021. CHIPS Alliance AIB-3D. https://github.com/chipsalliance/AIB-specification/blob/master/CHIPS_Alliance_AIB_Deep_Dive_AIB-3D_Draft_Spec_Review_0.9.pdf.

[61] Shin-Hua Chao and Chao-Fu Weng. 2016. Fine Line/Space IC Substrate Made by Selectively Fully Additive Process. In *2016 International Conference on Electronics Packaging (ICEP)*. IEEE, Hokkaido, Japan, 341–344. https://doi.org/10.1109/ICEP.2016.7486843

[62] Dylan Stow, Itir Akgun, Russell Barnes, Peng Gu, and Yuan Xie. 2016. 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*, Frank Liu (Ed.). ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/2966986.2980095

[63] Emil Talpes, Douglas Williams, and Debjit Das Sarma. 2022. DOJO: The Microarchitecture of Tesla's Exa-Scale Computer. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. IEEE, Cupertino, CA, USA, 1–28. https://doi.org/10.1109/HCS55958.2022.9895534

[64] Mark Wade, Erik Anderson, Shahab Ardalan, Pavan Bhargava, Sidney Buchbinder, Michael L. Davenport, John Fini, Haiwei Lu, Chen Li, Roy Meade, Chandru Ramamurthy, Michael Rust, Forrest Sedgwick, Vladimir Stojanovic, Derek van Orden, Chong Zhang, Chen Sun, Sergey Y. Shumarayev, Conor OKeeffe, Tim T. Hoang, David Kehlet, Ravi V. Mahajan, Matthew T. Guzy, Allen Chan, and Tina Tran. 2020. TeraPHY: A Chiplet Technology for Low-Power, High-Bandwidth In-Package Optical I/O. *IEEE Micro* 40, 2 (Jan. 2020), 63–71. https://doi.org/10.1109/MM.2020.2976067

[65] Dong Xiang, Bing Li, and Yi Fu. 2019. Fault-Tolerant Adaptive Routing in Dragonfly Networks. *IEEE Transactions on Dependable and Secure Computing* 16, 2 (March 2019), 259–271. https://doi.org/10.1109/TDSC.2017.2693372

[66] Dong Xiang and Xiaowei Liu. 2016. Deadlock-Free Broadcast Routing in Dragonfly Networks without Virtual Channels. *IEEE Transactions on Parallel and Distributed Systems* 27, 9 (Sept. 2016), 2520–2532. https://doi.org/10.1109/TPDS.2015.2503746

[67] Dong Xiang, Yueli Zhang, and Yi Pan. 2009. Practical Deadlock-Free Fault-Tolerant Routing in Meshes Based on the Planar Network Fault Model. *IEEE Trans. Comput.* 58, 5 (May 2009), 620–633. https://doi.org/10.1109/TC.2008.211

[68] Yuan Yao and Zhonghai Lu. 2016. Opportunistic Competition Overhead Reduction for Expediting Critical Section in NoC Based CMPs. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, Seoul, South Korea, 279–290. https://doi.org/10.1109/ISCA.2016.33

[69] Jieming Yin, Zhifeng Lin, Onur Kayiran, Matthew Poremba, Muhammad Shoaib Bin Altaf, Natalie Enright Jerger, and Gabriel H. Loh. 2018. Modular Routing Design for Chiplet-Based Systems. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, Los Angeles, CA, 726–738. https://doi.org/10.1109/ISCA.2018.00066

[70] Brian Zimmer, Rangharajan Venkatesan, Yakun Sophia Shao, Jason Clemons, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel S. Emer, C. Thomas Gray, Stephen W. Keckler, and Brucek Khailany. 2020. 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* 55, 4 (April 2020), 920–932. https://doi.org/10.1109/JSSC.2019.2960488