2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-2027-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/HPCA53966.2022.00091

Application Defined On-chip Networks for Heterogeneous Chiplets: An Implementation Perspective

Tianqi Wang[1,*], Fan Feng[1,*], Shaolin Xiang[1,*], Qi Li[1], and Jing Xia[1,**]

[1]*Huawei*

[*]*Contributed equally to this paper*

[**]*Corresponding author: Jing Xia, dio.xia@hisilicon.com*

ABSTRACT

With the help of advanced packaging technologies to integrate multiple chips (e.g., CPU, AI, IO), a chiplet-based SoC design process can enable fast system construction. However, the design of network-on-chip used within the individual chiplets and across chiplets is an extremly challenge. We introduce the design process and methodology of a bufferless multi-ring NoC for heterogeneous chiplet-based SoC. Our design is portable and can be used in diverse scenarios, like Server-CPU, AI-Processor, and Baseband-Processor.

The co-design of the application, architecture, and implementation is the key to make the system power efficient and high performance. We determined many architectural design choices by reflecting an analysis of a set of target applications by application teams and several physical implementation constraints provided by development teams. In this paper, we present the pragmatic practice of our co-design effort for the NoC. As a result, the system has been proven to achieve 16TB/s bandwidth in an AI processor and low latency, in a server CPU with nearly one hundred cores.

1. INTRODUCTION

Large monolithic System-on-Chips usually lead to poor yield and tremendous cost in verification effort and advanced manufacture. At the same time, some modules that do not need the advanced process nodes (e.g., PCIe controller) or do not scale well (e.g. IO buffers) can be moved to a separate chiplet manufactured in a mature technology node with lower costs and excellent yields. These trends drive the engineers to design the whole system in a chiplet manner [13]. This method tries to break a traditional monolithic SoC into several smaller chiplets, each of which can achieve better yield and become easier to be reused among different scenarios. Several works from academia and industry have shown that by using advanced packaging technologies, a chiplet-based SoC can tightly integrate multiple heterogeneous chips into a single SoC (e.g., CPU, GPU, HBM, FPGA) [9, 19]. Currently, it seems to be a promising way to tackle the failure of Moore's law and Dennard's scaling.

There are lots of research and engineering challenges associated with heterogeneous-chiplet based system, our work focus on the problem of network on-chip. There are several chiplet-based architectures have been released [4, 12, 18]. With the help of advanced packing technology, the area of the whole system is as large as **beyond a reticle**. Therefore, the interconnect fabric of the chiplet-based system needs to spread flits to larger physical space. Moreover, an SoC constructing from multiple heterogeneous chiplets will introduce more complexity and cause new cyclic dependencies among the chiplets. For example, even though each chiplet is properly verified, the fully integrated system still suffers from correctness issues [14, 21, 23, 24, 26].

A successful NoC design must make trade-off decisions from three options: application, architecture, and physical design, which means that it is impossible to have all three of these aspects optimized at the same time. We call the trade-off the trilemma of system design. This paper focuses on what we did in the co-design process and how we made several decisions on the NoC design. The technical challenge mentioned above and the high cost of developing a bug-free heterogeneous chiplets system drive us to try to reuse the on-chip network design across various application scenarios, such as wireless station, smart NIC. Server-CPU and AI-Processor are two thriving and prosperous application domains in the semiconduction industry, so we use these two as examples to demonstrate the NoC co-design methodology. This work makes the following contributions:

- We propose a highly scalable bufferless multi-ring NoC design for a heterogeneous-chiplet based system. We show this design can be effectively deployed in various scenarios, like Server-CPU and AI-Processor.
- We introduce the application-architecture-physical implementation co-design process and design methodology of the bufferless multi-ring NoC system. We show that for a heterogeneous-chiplet based system, distance per clock cycle is a more suitable metric for NoC co-design optimization.
- We show that the NoC design can achieve cache coherency among nearly one hundred cores (in one package) and low latency off-chip memory access in Server-CPU scenario. For AI-Processor, the same NoC design

978-1-6654-2027-3/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCA53966.2022.00091

provides 16 TB/s bandwidth. These key features are far beyond to other state-of-art implementation.

The organization of the rest of the paper is as follow: Section II presents the overview of the heterogeneous-chiplet NoC project. Section III presents the process and the methodology of the trilemma co-design. Section IV describes the NoC's architecture and implementation in server CPU and AI processor. Section V shows the experimental evaluation results of the system. Section VI concludes this paper with some retrospective comments.

2. BACKGROUD AND MOTIVATION

In the past two decades, the complexity of the SoC increase 1000 times, and the die size increase 5 times. The slowdown of Moore's Law has driven the advanced semiconductor manufacture process to become increasingly fragile and costly. Unfortunately, a countervailing trend - increasing processor design, verification, and management costs - is putting tremendous pressure on the scale of the SoC system. As a result, new design and package technology are being developed and commercialized where different chiplets can be connected using low-latency and high-bandwidth interconnect substrates. Therefore, a large SoC can be disintegrated into several smaller component chiplets and then reintegrated into a full system. [11] shows that chiplets can be integrated without significant performance degradation (less than 5%). Since smaller chip size can achieve better yield, chiplet-based SoC can decrease overall system cost. Moreover, the packaging technology also make it possible to combine a set of chiplets to build a heterogeneous chiplet SoC.

In a heterogeneous chiplet SoC, engineers can use the state-of-the-art manufacture process to build the critical components (like CPU core, ISP system) and implement some modules (like IO, Transceivers) with a mature technology node with lower costs and better yields. Furthermore, in this manner, several development teams can decouple their technology evolution roadmap. As shown in Figure 1, even though CPU, AI, Wireless, IO, and SoC technology teams can have different development pace, we can deliver products persistently with state-of-the-art technology. To integrate these components seamlessly, the SoC, especially the interconnect network, must be designed carefully. In the following session, we will give a brief introduction to our Lego-like SoC project and analysis the optimization targets of the NoC in different scenarios.

2.1 Lego-like SoC Project Overview

Lego-like SoC project tries to support several various application domains, like server CPU, desktop CPU, smart NIC, AI inference/training processor, and wireless station (shown in Figure 2). To achieve this target, we categorize these scenarios into several primitive chiplets. As Table 1 shows, these primitive chiplets (like AI-Die, Communication-Die, Computation-Die, and IO-Die) are highly specialized for their own compute pattern and are maintained by separated deveopment teams of domain experts. The Lego-like SoC project provides some basic components, like DDR controller, HBM controller, and cache coherency protocol module. Moreover, the most critical part of the project is the on-chip network,

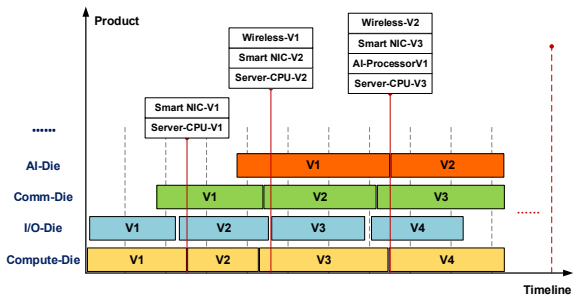

Figure 1: Technology evolution roadmap: decouple development, deliver products persistently

Table 1: Chiplet primitives' brief introduction

Chiplet Usage	Brief Description	Basic Componets
Compute Die	Provide general-purpose computing power	CPU-core, Cache
IO Die	Provide more IO extention	Transceivers, Ethernet
AI Die	Provide AI-realted computing power	AI accelerator, Cache
Commu Die	Support communication service	DSP, Protocol accelerators

which connects all these basic SoC components and various domain-specific process engines located in heterogeneous chiplets. In the rest of the paper, we use Sever-CPU and AI-Processor as examples to demonstrate the NoC design.

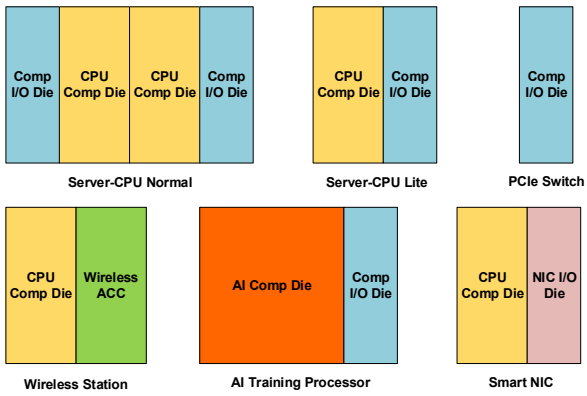

Figure 2: Lego-like chiplet

2.2 KPIs of NoC Design

As mentioned above, at the beginning of the project, we defined the Key Performance Indicators (KPIs):

1. Network Latency: For Server-CPU, datacenter applications usually demand microsecond-scale tail latencies and high request rates. Moreover, most applications, like web service, big-data framework, handle loads that have high variance over multiple timescales. Achieving these goals in a Server-CPU requires low memory access latency. Therefore we set the low average latency as the key design target for

Server-CPU scenario.

2. Network Bandwidth: As shown in Figure 3, the arithmetic intensity of AI applications is much higher than general-purpose applications. For AI-Processor, the high density of computing force requires high memory access bandwidth. Thus the NoC, which is used as the bridge between bandwidth producers and consumers, needs to provide enough bandwidth. Hence, the bandwidth of the NoC is set as the key performance indicator for AI-processor.

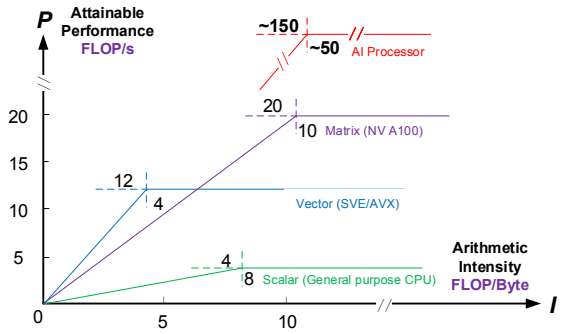

Figure 3: Roofline model for differnet application. The arithmetic intensity of AI is the highest.

3. Network Area Efficiency: From physical implementation view, the area of the on-chip network is also a critical metric. To spread flits over several chiplets, NoC has to involve more repeater and buffer to make sure timing closure. So the complexity and area of the NoC increase rapidly. Moreover, the NoC and related buffer cut the whole floorplan into smaller pieces, which introduce more trouble in other modules' placement and routing. So the area efficiency of the NoC is also treated as an important metric.

3. CO-DESIGN METHODOLOGY

As one of the most critical components in the SoC, the co-design of NoC is beyond traditional hardware-software co-design. To achieve the key performance indicators mentioned above, we have to make trade-off decisions from three aspects: application performance, architecture expressiveness, and physical implementation. As shown in Figure 4, it is impossible to have all three of the following at the same time:

1) Application performance: Traditional general-purpose Von-Neumman architecture cannot meet the computational demands of emerging applications. Embedding more domain-specific accelerator is a widely used approach to deal with this challenge, like integrating hardware GEMM engine to accelerate AI.

2) Architecture expressiveness: Although, domain-specific accelerators bridge the performance gap, still suffer from dealing with isolated programming interfaces and the expressiveness of the hardware architecture is becoming limited. Therefore, the effort to maintaining Von-Neumman architecture and shared memory model is trying to improve the hardware architecture's expressiveness.

3) Physical implementation: There ain't no such thing as a free lunch. Better performance and expressiveness mean more circuit complexity, which means more area, longer development time, and more labor cost to get timing closure. All these side effects will finally affect the commercial competitiveness of products.

In the rest part of this session, we present a detail analysis of these three aspects and provide an overview of the NoC co-design methodology.

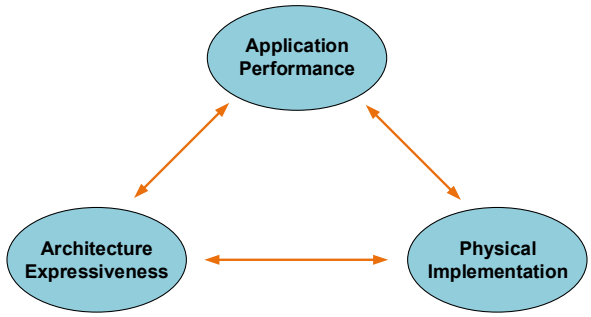

Figure 4: Impossible trinity

3.1 Target Application Domains for Co-Design

3.1.1 Server Applications' requirement

Under the co-design concept, the marketing, application software, and system software development team provide a set of target applications, as shown in Table 2, to guide the NoC design. The reason for choosing these applications as target applications are based on two criteria. One criterion is from the marketing viewpoint. General-purpose server-CPU is used in datacenter, cloud computing, or high-performance computing. Related applications require the processor can support a wide range of programs from web-service, database, to virtualization. Programs shown in Table 2 are widely used general-purpose benchmark and are basic software stack components of target consumers mentioned above.

The other criterion is from the viewpoint of the computational pattern. Server applications are diverse but they share a number of characteristics. First, the amount of load in form of client requests, processing, and communication resources are expected in a specified time period. Second, they compute on big data and the data follow the Zipfian distribution. Consequently, their long tails produce irregular requests and demand high-capacity memory. Third, server-based datacenter workload exhibit two major forms of parallelism across tasks and across data partitions, the parallelism in the system organization demands for the scale-out processors. Programs mentioned in Table 2 cover integer operation, floating-point operation. They also cover various data access patterns and problem sizes. In brief, they are all representative examples.

From NoC design aspect (shown in Table 2), typical scenarios for datacenter and cloud computing, like SPECint, SPECpower, Unixbench, and Glibc emphasis single-core performance and rarely require inter-core communication. Moreover, these benchmarks usually rely on pointer-based data structures and require plenty of off-chip memory access. Therefore supporting low latency off-chip memory access is extremely important for the general-purpose server-CPU NoC design.

Table 2: Typical benchmark for Server-CPU design

Bench-mark	Brief Description	Characteristics
SPECint	CPU integer processing power benchmark	Single core performance
SPECpower	Power and performance characteristics	Scalability of shared resource, JVM
Unixbench	Performance of a Unix-like operation system	Operation system, multi task
Glibc	Core lib for the GNU systems	UNIX Spec, POSIX, etc

Table 3: Typical benchmark for AI-Processor design

Neural Network	Application Domain	Basic Operator
ResNet	Image Classification	Convolution, Skip-connect
BERT	NLP	Transformers
Wide & Deep	Recommendation	Embedding, MLP
GPT	NLP	Transformers

3.1.2 AI related domains' requirement

During the co-design process, the marketing and application software team select several typical neural networks used as NoC design's guideline (shown in Tabel 3). From the viewpoint of marketing, these neural networks cover various market segments, like computer vision (ResNet), natural language processing (BERT), and recommend systems (Wide & Deep). Our AI-Processor needs to support ranging from tiny neural networks' inference (Yolo-v3) used in swing face detection to complex model's training (BERT, ResNet). From the viewpoint of computational patterns, these examples are representative in AI domain.

AI-Processors are used in various applications, such as natural language processing, autonomous driving, robots, smartphones, intelligent IoT devices, and etc. As shown in Table 3, AI-processors support ranging from tiny neural networks' inference used in swing face detection to complex models' (e.g. BERT, ResNet) inference and training. Typical operators of AI applications are 2D-convolution, vectorized compute, etc. The basic data structures used in AI are high-dimensional tensor and vector, pointer-based irregular data strucutres are rarely used. As mentioned above, AI applications have higher arithmetic intensity, more data reuse, and present obvious memory bandwidth hungry.

From NoC design viewpoint, typical neural network models (like ResNet, BERT, Wide & Deep, and GPT) size is as large as 175 billion, and require up to 3.14×10^{14} GFlop. The NoC of AI-processors act as the bridge between the high-density floating-point compute engine (bandwidth consumer) and high bandwidth off-chip memory (bandwidth producer). To take advantage of the applications' high data reuse attribute and fill the bandwidth gap between consumer and producer, NoC design must focus on bandwidth.

3.2 Architecture Expressiveness

More and more high-level programming languages are emerging. Their expressiveness are all inherited from their common ancestor - the von Neumann architecture. The von Neumann architecture has a number of characteristics that have had an immense impact on the programming language and existing algorithm. These characteristics include a single, centralized control, housed in the CPU, and a separate storage area, which can contain both instructions and data. Even though many widely used languages such as C, C++, and Java have ceased to be strictly von Neumann by adding support for parallel processing, in the form of threads. However, the **shared memory abstraction** is still has a significant effect on the expressiveness. In order to maintain the expressiveness of the software and hardware, **our architecture development team choose to stick to the shared memory abstraction.**

AMBA5 CHI protocol help engineers to build a multi-core cache coherence system and maintain the performance as the number of components and quantity of traffic rises. It provides high frequency, non-blocking data transfers and constructed with a layered architecture well suited for packetized on-chip networks. **Based on AMBA5-CHI, the NoC needs to be customized to support tens of large CPU-cores or AI-cores located on separated chiplets**.

3.2.1 AMBA5-CHI based Server-CPU Architecture

Figure 5(A) shows the memory sub-system hierarchy of the Server-CPU. Each CPU core has a private 64 KB L1 instruction cache, 64 KB L1 data cache, and 512 KB L2 cache. Moreover, the L3 cache system chose a hybrid design: L3 cache is partitioned into L3 tag cache (store tag to decrease snoop latency) and L3 data cache (store data to achieve high capacity). Four CPU cores are grouped as a cluster and these four cores share an L3 tag cache. The L3 data cache is instanced as a spearated module. From the L3 cache's view, the NoC provides the AMBA5-CHI protocol service to distributed L3-Tag/Data cache. As Figure 5(A) shows, only the L3 cache hit/miss event can invoke an NoC transaction. Moreover, the multi-level cache hierarchy can block most of the memory requests from CPU cores. Thus we can assume that any two L3 cache hit/miss are irrelevant in most cases. In other words, **each two NoC transactions are independent and stateless for Server-CPU scenario**.

3.2.2 AMBA5-CHI based AI-Processor Architecture

Figure 5(B) shows the architecture of the AI-Processor. The processing engine of AI-Processor is the AI-core, which contains three kinds of computing units (Cube, Vector, and Scalar). The cube unit can calculate a 16 x 16 matrix multiply in one cycle. The vector unit can handle various vectorized operators and other scalar operators are executed by the scalar unit. Each AI-core also contains 32KB instruction cache, 64KB L0-A/B/C cache used for store matrix for cube unit, and 1MB L1 cache used as a shared buffer for all three kinds of computing units. For multiple AI-cores, distributed L2 caches provide shared memory abstraction for programmers with the help of AMBA5-CHI protocol. Moreover, high

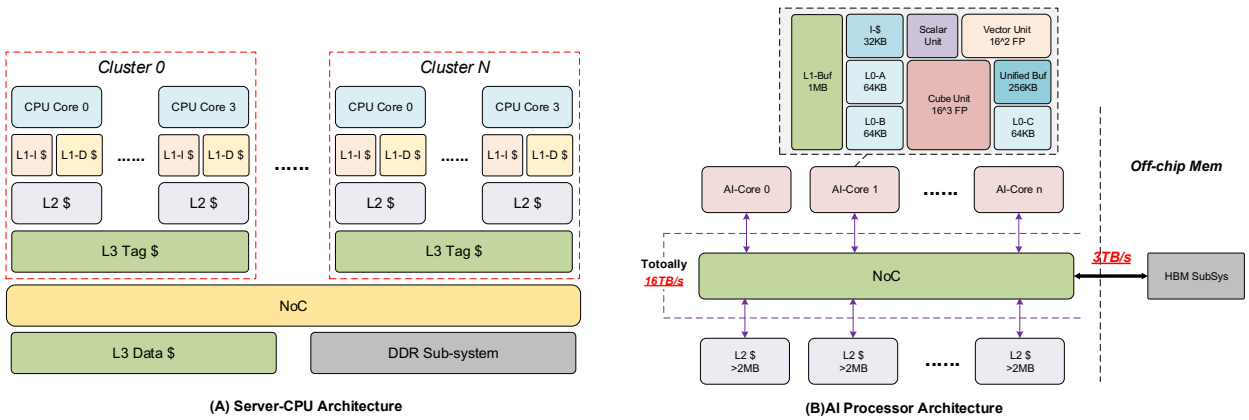

Figure 5: Server-CPU and AI-Processor Architecture Overview

bandwidth (500GB/s x 6) HBM modules are used to provide efficient memory capacity. As mentioned above, the AI-core provides an extremely high-density floating-point compute force (16x16x16 for cube units, 16x16 for vector units). According to the arithmetic intensity of AI applications, the bandwidth requirement of distributed L2 cache is larger than 10TB per second. Therefore the NoC need to provide enough bandwidth to compensate for the bandwidth gap between HBM, L2, and AI-cores. The workloads of AI-cores are various, thus the traffic flow of NoC may change dramatically. Thus the equilibrium of the NoC is quite important. Thus the architecture team chooses to associate the L2 cache in a **interleaved manner**, so that the memory access request traffic can be evenly spread across the whole chip. From NoC's perspective, AI-cores' each L2 cache request can invoke an NoC transaction. Most of the memory requests are sequential and the interleaved L2 cache makes sure that **each two NoC transactions are independent and stateless for AI-processor scenario**.

3.3 Physical Implementation Constraints

Our physical implementation team also provide important suggestion in the NoC co-design process. From the perspective of circuit physical implementation, an on-chip network consists of metal wire fabric, which is the most important component to transport flits over heterogeneous chiplets, and all necessary buffer and logical circuit, which is attached to make the NoC functional. However, metal fabric, logical circuit, and memory are implemented with different technology on different layers during semiconductor manufacture. Moreover, wires have parasitic resistance and capacitance which increase with the length of wires. To meet a specific target frequency (3GHz), a long wire needs to be split into several segments, and repeaters must be inserted between the segments. Table 4 shows the key parameters of two NoC wire metal fabric implementations. As shown in the table, the Mx My layer-based high-density metal fabric has smaller width and pitch size, and the flits are transmitted shorter distances in one clock cycle. Although the My layer-based high-speed metal fabric has a larger width and pith size, the flits can travel a longer distance in one cycle. As shown in Figure 6, the high-density wires are nearly continuous metal and can not be placed over any other circuit. However the high-speed wires only occupy intermittent regions, and SRAM blocks can be placed in these slots.

From NoC design viewpoint, the on-chip network needs to spread flits across agents located in distanced chiplets. The wire metal fabric and other necessary circuits (logic and buffer) are implemented in different layers. Each hop in the NoC demands a cross station which will consume more silicon area and introduce more circuit complexity. That makes time-closure more challenging. Even though the metal fabric of high-density wire can save area, but the limited jump distance will cause more area consumption for cross stations. And the metal fabric can not be placed over other circuits. Thus the NoC cut the whole floorplan into separated regions which make the place-and-routing more difficult. Contrastly, the high-speed wire is much more area efficient and the stride slot can be used for SRAM. Thus it is a better choice for NoC. In one word, **distance per cycle is a suitable metric** and a simplified circuit structure is more friendly for physical optimization.

3.4 Bufferless Multi-Ring NoC

As mentioned above, several development teams provide highlights from their viewpoint. The requirements for the NoC are listed:

- Application: The NoC must provide high bandwidth (larger than 15TB/s) and maintain low latency for cross-chiplets requests at the same time.
- Architecture: The architecture of NoC should be flexible. Each two NoC transactions are independent and stateless.
- Physical Implementation: NoC design prefers to a simplified circuit structure and tries to improve distance per cycle.

3.4.1 Heterogeneous-Chiplet Consideration

As mentioned in Section 2.1, our products cover several various application domains based on the combination of heterogeneous chiplets.

A chiplet is a relatively tightly coupled system. However, transceivers, ethernet, and domain-specific accelerators that

Table 4: Physical Implementation Key Parameters

Type	Metal	Width	Pitch	Bus Width	Jump 3GHz	Stride	Over
High-dense wire fabric	Mx-My	x1	x1	x1	600um	0um	Nothing
High-speed wire fabric	My	x3	x3.5	x2.5	1800um	200um	SRAM

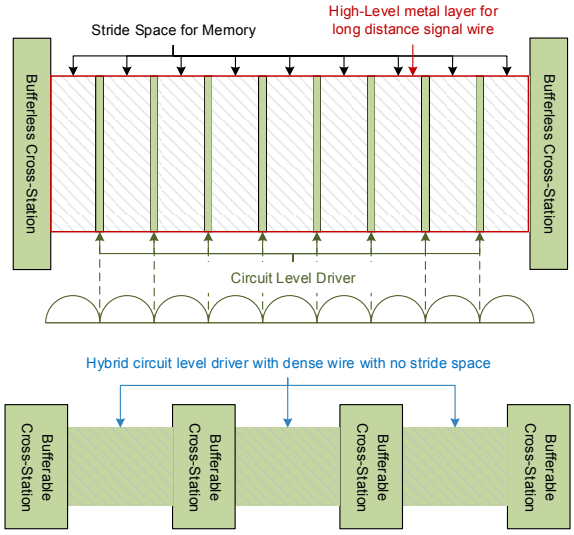

Figure 6: NoC Physical implementation constraints: high-dense wires save area but have limited jump distance. High-speed wires cost more space, jump further and the stride slots can be used for SRAM.

are located on the non-compute die have frequent communication between compute die. To meet the independence and interactivity of different chiplets, ease the flow control, and achieve deadlock avoidence, a ring based on-chip network with the carefully designed cross-point between different rings on different chiplets is adopted.

According to the requirements of development teams and heterogeneous-chiplet considerations, our choice is bufferless multi-ring NoC. The qualitative discussion of the advantages and disadvantages of this design choice compared to traditional buffered schemes follow next:

3.4.2 Advantages

No buffers is the key advantage of this approach. It helps us reduce both circuit complexity and energy consumption. Since bufferless NoC eliminates buffer, there are no virtual channels and there is no need to manage buffer allocation/deallocation. The simplified circuit structure can achieve higher clock frequency and less area consumption with limited physical implementation effort. Compare to the communication-based flow control mechanism used by the buffered routing scheme, bufferless NoC uses purely local and simple flow control without any need for communication between routers.

Compare to the single ring scheme, the bufferless multi-ring structure has good scalability and can connect heterogeneous chiplets with simple cross-ring extensions. Moreover, bufferless multi-ring NoC can decrease average latency when the number of agents rises. This is extremely important for the scenario like server CPU or wireless station.

3.4.3 Disadvantages

As shown in [16], one of the key downside of bufferless NoC is that it can increase average latency because of deflection routing. Also, the bufferless method will reduce the available network bandwidth as all in-network flits consume wire fabric resources. To deal with these problems, we implement an ejection-tag mechanism to make sure the flits can not defect more than one lap in the bufferless multi-ring. Section 4.1 provides details of this mechanism and experiments show the performance of our design is better than state-of-the-art implementations.

Bufferless NoC requires that header information be transmitted with each flit because each flit of a packet can follow various paths. This downside introduces additional overhead. As mentioned above, for Server-CPU and AI-Processor, the granularity of the NoC transaction is as large as one L2/L3 cache line. Thus the area saved by simplified bufferless cross station is far greater than the additional header information's consumption.

Since bufferless NoC can deflect individual flits, flits of a packet can arrive out-of-order and at significantly different points in time at the destination agent. This effect can increase destination-side buffering requirements. However, as mentioned above, our architecture ensures that each two NoC transactions are independent and stateless. Thus make each transaction as a single flit attached necessary header information is a reasonable choice. And the AMBA5-CHI is a non-blocking and out-of-order protocol that require buffers in each node. The out-of-order mechanism is bufferless NoC friendly and the necessary buffer can be reused by the NoC.

Livelock and service starvation are potential challenges in bufferless NoC. In section 4.1, we provide a scheme to solve the problem. Deflection-based routing of bufferless-ring NoC is deadlock-free. For multi-ring, deadlock only occurs in the cross-ring scene. This paper shows a SWAP mechanism embedded in the cross-ring bridge can make the multi-ring deadlock-free (shown in section 4.4).

During the co-design process, we believe that these disadvantages have limited effect in our scenario or can be solved with mechanism extensions. More detail about the architecture and implementation will be introduced in the next section.

4. ARCHITECTURE AND IMPLEMENTATION

This section shows the architecture and implementation detail of the NoC. Firstly, this section presents the basic components of the on-chip network. Then we introduce how these parts are integrated as NoC for heterogeneous-chiplets Server-CPU and AI-Processor. At last, we provide a

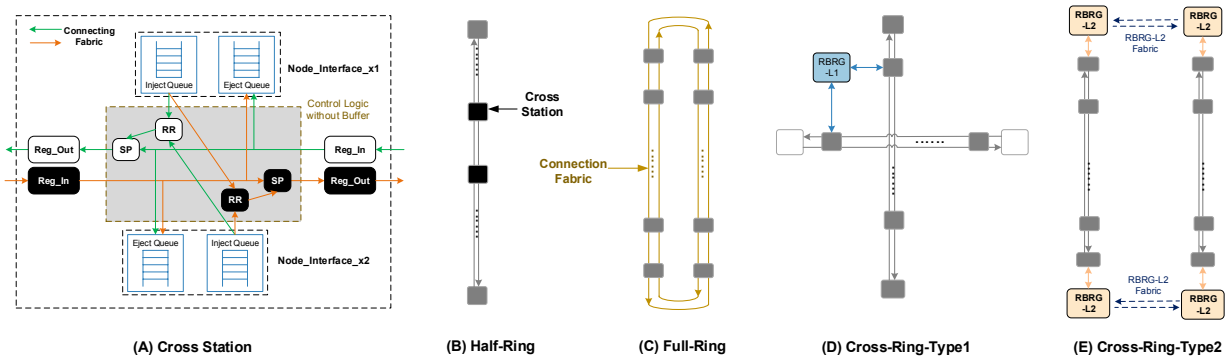

Figure 7: NoC basic components: (A) Cross station used for node connection. (B-C) Half/Full-Ring used for various node quantity and latency requirements. (D-E) Two types of cross-ring for intra- / inter- die communication.

SWAP mechanism to guarantee the NoC is deadlock-free for heterogeneous chiplets.

4.1 Common Components of NoC

Cross stations and connection fabrics are the basic elements that constitute a ring. Cross stations crossover the connection fabric as Figure 7(A) shows, and each of them can connect up to two devices through the node interfaces. Each node interface has an Inject Queue that can inject flits to both directions of the ring, and similarly, an Eject Queue that can receive flits from both directions. Since each flit in the Inject Queue can access to both directions, the cross station also needs to carry out ring selection, this is done by a straightforward approach to achieve the shortest routing path according to the source and destination address of the flit.

4.1.1 Flits priority control

With the control logic, a cross station is in charge of the routing of the on-the-fly flits that pass through the station, the injection of new flits from both nodes, and the ejection of flits to both nodes. As Figure 7(A) shows, the existing on-the-fly flits have the highest priority and the new injection flit from the node interfaces is selected by the manner of round-robin (RR).

4.1.2 Livelock-free and starvation-free

Guaranteeing the absence of starvations and livelocks is extremely important for our bufferless-ring based system. If a flit unable to obtain a ring slot for a certain cycle because of competition failure with the on-the-fly flits, an I-tag (Inject-tag) will be attached to the current slot, which means that the slot is reserved for the "defeating" flit. A slot with I-tag won't be occupied by other flits even if it's free, after the slot turns back to the cross station with a successful injection, the I-tag is removed.

Similar to injection failure, flit deflection happens when the flit arrives at the destination but there is no free space in the Eject Queue, the flit has to pass through the cross station. In this situation, an E-tag (Eject-tag) is created, when there is a free buffer in Eject Queue, it is reserved for the deflection flit. When the deflection flit ejects to the reserved buffer, thereupon the E-tag is cleared.

4.1.3 Ring bridges and multi-rings

There are two cases of the multi-ring system. The first one is a vertically and horizontally interwoven network composed of multiple rings intra-chiplet (Figure 7(D)). The second one is the connection of rings inter-chiplet (Figure 7(E)). Additional components are needed to support cross-ring transportation. In the first case, ring bridges (RBRG-L1) act as "devices" that reside in every intersection. RBRG-L1 is used for data buffering for the flits that need to exchange a ring path, routing information generation, as well as potential asynchronous processing. In the second case, we use a second-level ring bridge (RBRG-L2), which is the inter-chiplets communication component, to achieve the connection between different Dies, RBRG-L2 has similar buffering and routing info-generation functionality but is more complex than RBRG-L1. Besides providing backpressure flow control, RBRG-L2 also takes charge of deadlock resolution that will be introduced later. As to the physical layer of inter-chiplet interconnection, we use an in-house developed low latency parallel IO block with a specific protocol cross-chiplet fabric, to transmit data packets between chiplets.

The rings of the system can be categorized into "half-ring" and "full-ring", a ring with a single clockwise loop is called half-ring (Figure 7(B)), and a ring with bidirectional loops (clockwise and counterclockwise) is referred to as the full-ring (Figure 7(C)). The full ring can provide more node connection in the case of the same side length of a chiplet, and thus provide higher capacity and throughput at the cost of hardware area.

4.2 Server-CPU's NoC

Server-CPU's NoC design is shown in Figure 8 (A). With the basic components and routing mechanism of the ring, the next step is to attach other components to the ring to form an independent chiplet.

As mentioned above, our server CPU is designed for optimizing access latency. With this principle, devices such as CPU clusters, DDR controllers, L3 cache, and last level cache (LLC) that demand for high frequency and low latency are attached to in the CPU Compute Die (CCD). More specifically, the NoC in the CCD is a full ring.

For the latency-tolerance devices, such as PCIe, Ethernet

interface, hard disk interface, and various service-related accelerators, they are integrated into the I/O Die with a half ring. Apart from the functionalities and components described above, I/O Die provides the scale-up ability that is a feature of critical importance in server CPUs via PA (Protocol Adapter), which is an interconnection module with several SerDes links for inter-chip data access across chips.

Those basic components along with the ring constitute independent Dies logically, and multiple Dies can be connected to form a larger chip through RBRG-L2. With the multiple SerDes links on the I/O Die, we can scale the chip up to a 4P (4 chips) system with a total core number of more than 300 and maintain cache coherence.

4.3 AI-Processor's NoC

Computation capacity and aggregate bandwidth are the first considerations to the training systems, due to the massive AI models with huge parameters. These requirements are pushing the limits of today's training systems. A straightforward solution is to integrate as many devices (i.e., AI cores) as possible on the chip, but it requests an efficient NoC. Mesh or mesh-like NoCs have the desired scalability, but traditional mesh-like NoC confines the devices to the intersection of the mesh, which may weaken its advantages. To overcome the scalability barrier, we can build the NoC in another way.

The NoC in our AI training processor is a multi-ring based mesh where RBRG-L1s take responsibility for cross-ring transportation as described in section 4.1. With the dedicated cross-ring devices, each ring has a large number of ring stops that are not restricted to the number of intersections. By doing so, the total devices connected to the NoC are maximized with a reduced overall number of rings.

Different from the L3 cache in the server CPU, the distributed L2 in the AI processor only provides data storage, and the functionality of set-associative cache is implemented by the LLC. So all data requests initiated by the AI Core are first received and processed by LLC (Path 1 in Figure 8(B)). When the LLC gets a directory hit, data can be transferred between L2 and the AI Core (Path 2 and Path 3 in Figure 8(B)), while when the directory miss, L2 requests data from HBM through LLC, and the data is transferred from HBM to L2 (Path 4 in Figure 8(B)). Communication between AI core and L2 and communication between L2 and HBM are the most significant on-chip traffic flow. **Moreover these two traffic flow is orthogonal**.

Among the communication patterns, an AI core will only have an access request to data blocks but never initiate a request to the other AI cores, similarly, an L2 will never access the other L2, which means that the communication is somewhat limited and localized by their access patterns. Based on this observation, we put all the AI cores on the vertical rings and the memory-related nodes on the horizontal rings as shown in Figure 8(B). In this way, any request on the routing path takes no more than one ring change to reach the destination node. It means that we can use an X-Y or Y-X routing algorithm according to their source and destination information.

The balanced layout of a large number of devices and the simple routing algorithm is the key to achieve the desired high computation capacity and high bandwidth of the processor.

Similar to the server CPU, the AI Compute Die can connect to I/O Dies through the RBRG-L2 nodes for further scale-up and I/O interface expansion.

4.4 SWAP based Deadlock-Free

As mentioned in Section 4.1, I-tag and E-tag ensure the flits on their destination ring can eject successfully to avoid starvation and livelock, while in the situation where flits interlock dependency is established between multiple rings, a deadlock resolution mechanism is needed.

Multi-ring based system introduces a new kind of deadlock between multiple rings within a chip, as shown in Figure 9. Figure 9 illustrates that a pair of rings with every flit wants to transmit to the other ring through the ring bridge, when there are no free slots on both rings, the transmission is stalled.

Escape virtual channel is a widely used recovery technique to break deadlocks without routing or injection restrictions, which may cause throughput decrease under normal non-deadlock circumstances. But additional slot reservation for escape virtual channel will inevitably increase network latency, so in the latency-sensitive Server-CPU scenario, we use a latency-friendly "SWAP" mechanism to break the deadlock.

Deadlock detection and deadlock resolution: In our multi-ring system, a deadlock is considered to be formed if the RBRG-L2 attached cross station consecutively fail to inject flits over a threshold cycle, which indicates that the ring slots, the Eject Queue, and the available Tx buffers in the RBRG-L2 are all occupied by cross-ring flits, consequently, the header of the Inject Queue cannot fulfill injection.

Once a deadlock is detected, the RBRG-L2 will be triggered to enter into deadlock resolution mode (DRM). During the DRM, reserved Tx buffers in RBRG-L2 are activated for deadlocked flit, a flit (Flit-B5) in the Eject Queue is pushed to the reserved Tx buffer, thus freeing space for a cross-ring flit. At the cycle when the traversing flit (Flit-B6) ejects to the vacant Eject Queue, the header in the Inject Queue (Flit-A1) takes Flit-B6's place to move forward on the ring.

With the support of simultaneously injection and ejection in the cross station, the header flit of Inject Queue and the traversing flit is swapped, in this way, flits can flow and finally reach their destination node. Valid flits flow implies that the deadlock is broken.

After the occupied Tx buffer is lower than a threshold, the deadlock is regarded recovered, and the RBRG-L2 and the corresponding cross station exit the DRM.

5. EXPERIMENTAL EVALUATION

5.1 Server-CPU Experiment Setup

Evaluation Platform: As mentioned in Section II, our design consists of two Compute-Die and two IO-Die. To run a wide variety of complex workloads efficiently, we have developed the whole system's RTL code and use a hardware circuit emulator to speed up simulation. GCC-7.2.1, OpenJDK-11 are used as the compiler. The version of C libraray is Glibc-2.1.7.

Server-CPU Benchmark: Selected benchmark for server-CPU NoC and AI-processor NoC are listed as follow:

- NoC bandwidth: LMBench is a micro-benchmark suite designed to focus on the basic operating system and

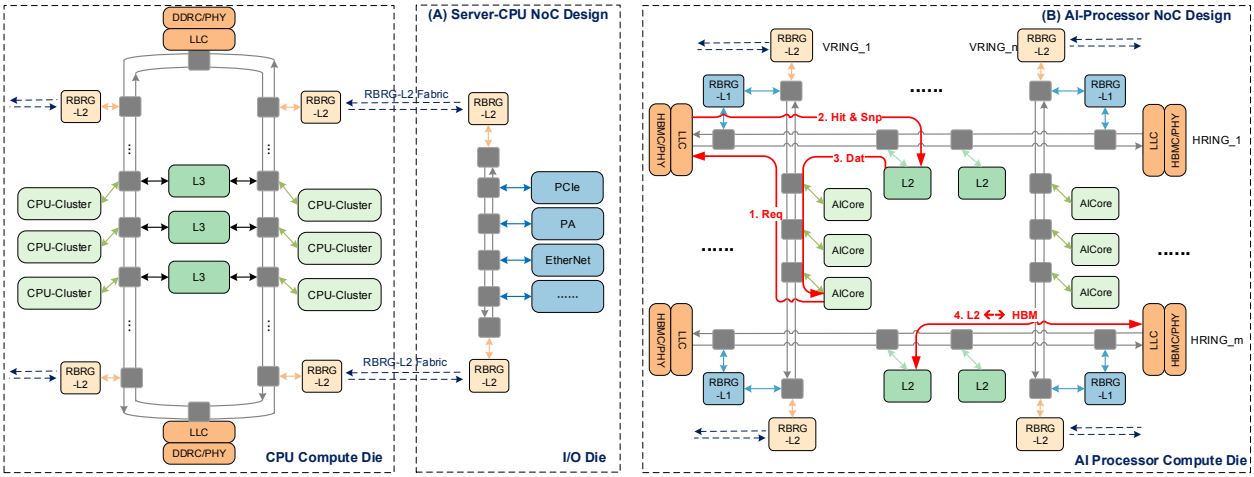

Figure 8: Server-CPU and AI-Processor NoC architecture: (A) Low latency interconnect specified for Server-CPU. (B) High bandwidth equilibrium NoC for AI-Processor.

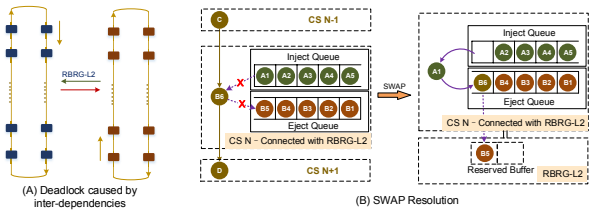

Figure 9: SWAP mechanism for deadlock-free cross-ring communication

hardware system metics. Part of LMBench is used to measure the NoC's bandwidth.

- NoC Latency: We build a test case to evaluate NoC latency: disable all L1/L2 cache and measure one CPU core's DDR access latency while all other CPU cores issue read, write, read/write request at various ratio. Another experiment is set up as follows: disable all L1/L2 cache, Core-0 changes 3MB data (located in the L3 cache) into modified/exclusive/shared status, then Core-1 access the same data and measure latency.
- Real Application: SPECint-2006/2017 and SPECpower-ssj-2008 are used as real application benchmarks.

System and Comparison: For real applications, like LMBench, SPECint, and SPECpower, we choose state-of-the-art server CPU (Intel-8280/Intel-8180/AMD-7742) as the baseline. For the experiments about NoC latency, we choose Intel-6148 as the baseline because its latency feature is better than Intel-8280 and AMD-7742. Our experiments provide single-core and one package's (contain multiple chiplets) result. To give a fair comparison among various NoC, we also scale down our system to baseline products and provide related benchmarks' results.

5.2 AI-Processor Experiment Setup

Evaluation Platform: AI-Processor: As mentioned in Section II, the AI-Processor consists of one AI-Die and one IO-Die. We developed a cycle-accurate software simulator for the AI processor. Tensorflow-2.3.0 is used to deploy neural networks.

AI-Processor Benchmark:

- NoC bandwidth: We use AI-processor's instruction trace record as NoC's input and insert several probes. Then we measure the NoC's bandwidth and equilibrium.
- Real Application: We choose a part of MLPerf Benchmark's training cases: ResNet-50 v1.5, Mask R-CNN, and BERT. For ResNet-50, ImageNet is used as the dataset and the training quality target is set as 75.90%. For Mask R-CNN, COCO is used for training and 0.377 Box min AP is set as the target. For BERT, we use Wikipedia-2020-01-01 as the dataset and set training quality target as 0.712 Mask-LM accuracy.

System and Comparison: In AI-Processor scenario, our work's and state-of-the-art baseline (NVIDIA-A100) vary greatly. Thus we only provide real neural networks' end-to-end experiment result for comparison.

5.3 Server-CPU Experiment Result and Analysis

Server-CPU NoC Bandwidth: Figure 10 shows the LMBench bandwidth over different baseline systems. All bandwidth related benchmarks are listed, like rd (memory reading and summing), frd (reading and summing of a file via OS's read interface), cp (memory copy), etc. To be fair, the experiment result normalizes the number of DDR4 channels and the frequency of DDR4. As shown in Figure 10, when single-core can occupy the whole package's DDR4 bandwidth, our work is 3.23 times better than Intel-8280 and 1.77 times better than AMD-7742 on average. When all CPU cores in the package are competition for DDR4 bandwidth, our work's bandwidth utilization is 1.19 times better than Intel-8280 and 1.7 times better than AMD-7742 on average.

Server-CPU NoC Latency: Table 5 shows the comparison of NoC access latency. As mentioned above, we tested the access latency of various cache status data in an intra-

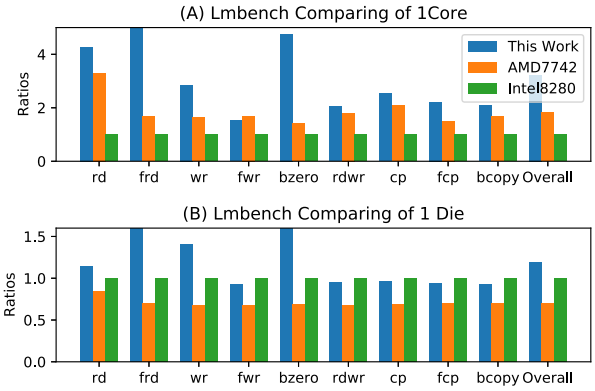

Figure 10: LMBench - NoC Bandwidth

Table 5: Inter-/Intra- chiplet access latency experiment result

	Status	This work	Intel-6248	AMD-7742
Intra Chiplet	M	44	NA	138
	E	44	NA	139
	S	48	NA	139
Inter Chiplet	M	65	91	140
	E	65	91	138
	S	69	91	138

or inter-chiplet scenario. In most cases, our work has lower access latency. Moreover, Figure 11 shows the NoC latency competition experiment results. As mentioned above, one core's DDR4 access latency is measured while others compete for NoC resource. The horizontal axis indicates the time ratio of background read/write request traffic and the vertical axis represents the test core's access latency. Compare to Intel-6148, the turning points of this work come later in the case of a read, write, or hybrid background traffic noise.

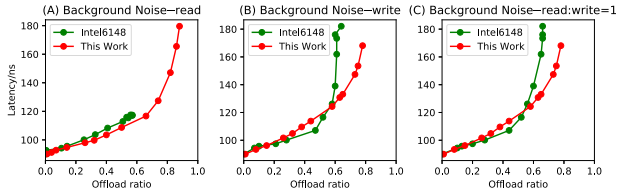

Figure 11: DDR latency competition experiments: increasingly background noise cause latency to increase sharply after turning points.

Server-CPU Real Applications: Figure 12 shows the experiment results of SPECInt-2017 (all results are normalized). In Figure (A), we compare the performance of single CPU core as a reference. And Figure 12(B) shows the comparison of one package's performance. To be fair, we also scale down our system to the same number of cores as Intel-8180 (shown in Figure 12(C)) and AMD-7742 (shown in Figure 12(D)). Similarly, Figure 13 shows the results of SPECInt-2006.

SPECpower is used to evaluate the power and performance characteristics of single server and multi-node servers. The score comparison is listed in Table 6. This work's score is

Table 6: SPECpower-ssj-2008 score comparison

Platform	1 Core	1 Package (Normalized)
This Work	134484.0	102984.5
Intel-8280	123911.0	86519.3
AMD-7742	129890.0	93196.1

Table 7: AI-NoC Bandwidth Test

R-W Ratio	NoC Bandwidth TB/s			
	Total	Read	Write	DMA
1:1	16.0	7.3	7.1	1.6
2:1	13.9	8.2	4.1	1.6
4:1	12.4	8.8	2.1	1.5
3:2	15.4	8.4	5.5	1.5
1:0	11.2	9.5	0	1.7
0:1	10.0	0	8.4	1.6

1.08 times higher than Intel-8280 and 1.03 times higher than AMD-7742 in single core test. Then the score of one package is also 1.19/1.11 times higher. Clearly, our work can achieve better score in each benchmark mentiond above.

Server-CPU NoC Analysis: Intel-8280 doesn't choose chiplet method and use a bufferless ring to build the multi-core system (28 cores). AMD-7742 uses heterogeneous chiplets and a switch located on IO-Die is used to connect four homogeneous CPU dies (64 cores). Compare to these two baselines, this work is more heterogeneous than AMD-7742, which may cause a worse inter-chiplet deadlock problem, and integrate much more cores (nearly one hundred). However, with the help of carefully designed bufferless multi-ring NoC, this work has less DDR access latency and achieves better scalability in various benchmarks.

5.4 AI-Processor Experiment Result and Analysis

AI-Processor NoC Bandwidth: Because of the diversity of AI tasks, the layers of DNN models are highly heterogeneous in operation and shape. Thus the diverse read/write ratio of the AI cores highly affect the performance of NoC. According to the various memory access behavior of diversified neural network layers, we build several traffic-flows with different read/write ratios to test the bandwidth of NoC. In the AI processor, two kinds of traffic patterns competing for the NoC bandwidth, i.e., read/write between AI-core and L2, and the data transportation between L2 and HBM via system DMA. As shown in Table 7, for typical read/write ratio, our NoC can provide more than 10TB/s bandwidth.

For AI-Processor, the NoC needs to distribute bandwidth across all AI-cores located on heterogeneous chiplets. Thus the bandwidth equilibrium is quite important. Therefore we integrated several probe in the NoC and the results are shown in Figure 14. Clearly, during the whole simulation process, the bandwidth distribution is very balanced among these monitors. For most of the time, all probes can get more than 80% of the maximum bandwidth.

AI-Processor Real Applications: Table 8 choose part of MLPerf Benchmark's training cases as demonstration. We choose TensorFlow-2.3.0 as the software framework for both platforms. To be fair, all experiments are based on FP16

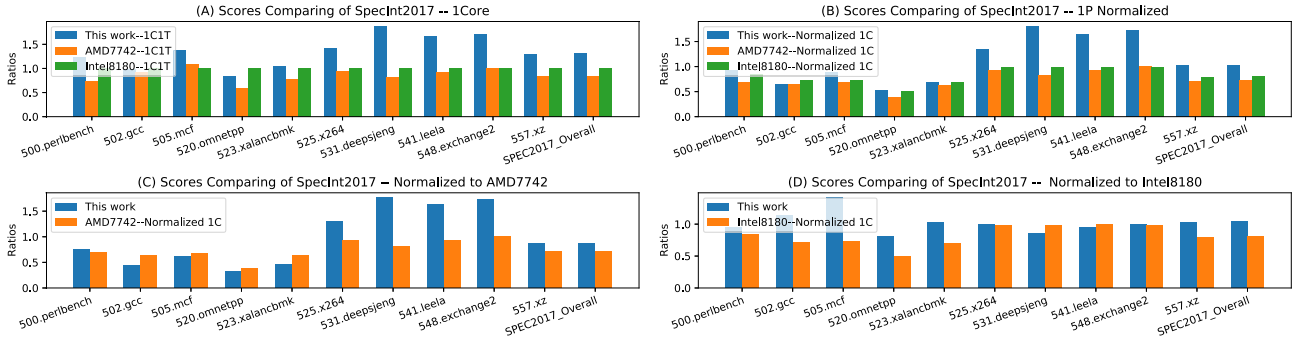

Figure 12: SpecInt-2017 Result

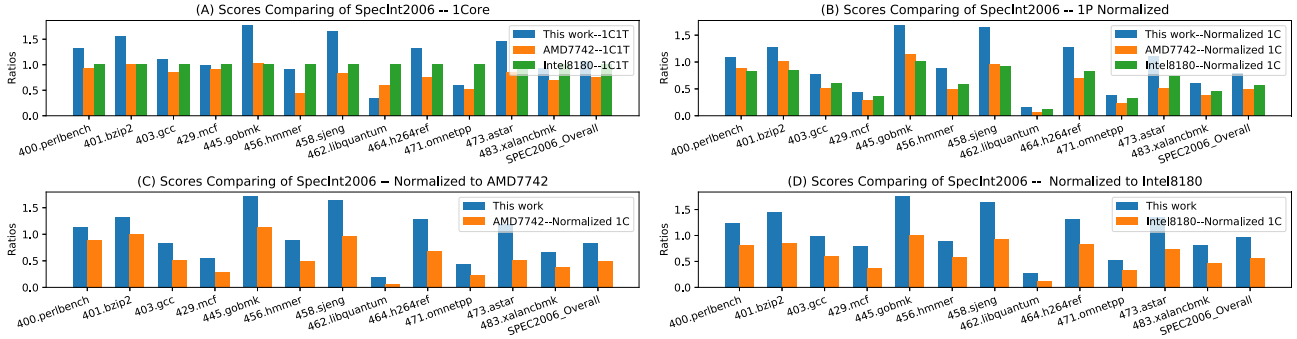

Figure 13: SpecInt-2006 Result

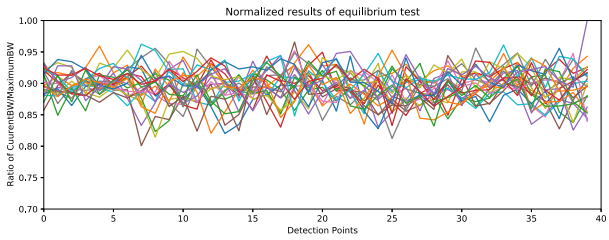

Figure 14: NoC bandwidth equilibrium experiment

Table 8: AI-Neural Network Performance and Energy Efficiency Comparison

	NV-A100 Perf/Energy	Ours Perf	Ours Energy
ResNet-50	x1	x3.2	1.89
BERT	x1	x2.99	1.50
Mask R-CNN	x1	x4.13	NA

mixed precision. As shown in Table 8, with the same training quality target, our work's performance is 3.2 times better for ResNet-50, 2.99 times better for BERT, and 4.13 times better for Mask R-CNN.

AI-Processor NoC Analysis: This carefully designed NoC can be reused in AI-Processor, which is a widely different scenario and achieves 16TB/s bandwidth. Moreover, this bufferless multi-ring NoC also provides a balanced bandwidth supply to all AI-cores. In several benchmarks, this AI-Processor can get better performance and energy efficiency than NVIDIA-A100.

6. RELATED WORK

The advanced 2.5-D and 3-D integration technologies have made chiplet assembled system an alternative way to monolithic chip design. Leading enterprises take actions in this area early for cost and scalability consideration, and industrial preference is a direct reflection of chiplet based system superiority. AMD use up to four Zeppelin chiplets [4] to build the first generation EPYC processor (Naples), and evolved to up to eight chiplets using a central IO Die to distribute the system level interconnects with infinity architecture in the following product family (Rome and Milan) [18]. Intel raised their heterogonous chiplet integration plan since they acquired Altera in 2013, and they release products like Intel Agilex FPGA and Xe HPC integrating heterogeneous chiplets with EMIB and Foveros integration technologies [9]. Different from the above processor build with an organic substrate, passive interposer, or silicon bridges, INTACT [25] presents the first measured on silicon active interposer, which integrating power management, distributed interconnects, and enabling an innovative scalable cache-coherent memory hierarchy, for a 96-core processor with six chiplets.

Table 9 shows the NoC related features of several state-of-the-art commercial processors. Since the IC process is slowly but continuously shrinking, enabling more cores in a single chip, how to interconnect cores becomes a key factor in leveraging many-core performance. The new 3rd generation Intel Xeon scalable processor (codename: Ice Lake-SP, ICX) [20]

Table 9: State-of-Art Commercial Processor NoC Works

	Core Count	NoC Intra-Chiplet	NoC Inter-Chiplet	Buffering stratagy	IC Process	Integration	Die Area (mm^2)
Intel Ice Lake-SP	40	Mesh	—	Bufferless	Intel 10nm	1 Die	640
Intel Sapphire Rapids	56	Mesh	UPI	—	Intel 7nm	EMIB	—
AMD Milan	64	Bi-Directional Ring Bus	Switched mesh	Buffered	TSMC 7nm	MCM	1008
AMD Instinct MI200-series	8 ACEs	—	Bi-directional rings	Buffered	TSMC 6nm	2.5 D Elevated Fanout Bridge	—
Fujitsu Fugaku	52	Ring Bus	Tofu-D over CPU nodes	Buffered	TSMC 7nm	CoWoS	—
Ampere Altra MAX	128	CoreLink CMN-600	—	Buffered	TSMC 7nm	1 Die	—

evolved their bufferless mesh within a chip based on their former generation server processors, the largest die of the series, ICX XCC, come with 40 cores due to the massive die size of about 640 mm^2. Intel's strategy of monolithic dies is a dead-end one. ICX dies are reaching the reticle limit. Therefore, Intel is currently moving to a multi-tile design and adopting 2.5D EMIB (Embedded Multi-die Interconnect Bridge) integration with their Sapphire Rapids. Sapphire Rapids [5] combines up to 4 tiles to achieve greater core counts. Although final details are not public yet, preliminary information indicates that each tile has 16 cores for the top-end model and Intel continues to employ a mesh interconnect for the CPU cores and other blocks within a tile. AMD is one of the pioneers to adopt chiplet design, the latest 3rd generation Epyc processor (codename: Milan) [7, 15, 17] integrate one IO Die and up to eight CCDs (compute die) with MCM packaging. AMD Milan uses a bi-directional ring bus inside the eight-core CCX (core complex) and an Infinity Fabric-based switched mesh to connect IOD and CCDs. Besides their Epyc-series server chips, AMD also announced Instinct MI200-series GPUs [1] that targerts supercomputers. The chip consists of eight ACEs (asynchronous compute engine) and comprises two identical die lying side by side, which are connected by a dedicated Infinity Fabric bus. For a four-chip cluster, eight chiplets are connected by additional two bidirectional rings. Fujitsu Fugaku [29] is an ARM-based server processor, it uses a ring bus-based NoC to connect CMGs (Core-Memory Group) in a chip and uses "TofuD" to build up a multi-chip system. Other ARM-based processors, such as Ampere Altra series and Alibaba Yitian 710, use Arm CoreLink Coherent Mesh Network to build up their SoCs.

To keep pace with the increasing core counts and provide efficient interconnection in the many-core chiplet-based system, several interconnection innovations have also been made to improve the scalability and reduce hardware cost. [3, 6, 8, 10, 16, 27] propose bufferless with deflection routers to reduce the area and power consumption of network buffers. And they expand the bufferless NoC by hierarchical rings. [2] proposed routerless NoC eliminate the costly routers by selecting a set of loops to connects cores and they improve the methodology by deep reinforce learning design to further optimize the NoC throughput, latency, and power. [28] introduce a simple, modular methodology for ensuring deadlock-free routing in multi-chiplet systems focusing on systems combining chiplets on an active silicon interposer. [22] provide an mechanism for enabling a blocked packet to move forward unless the buffer at the next hop is guaranteed to be free.

7. CONCLUSION

This paper presents a bufferless multi-ring NoC, which can be used to connect heterogeneous chiplets. Moreover, we retrospect the co-design process of the on-chip network. During the co-design, several development teams analyzed a set of target applications and make several architectural design decisions under several physical implementation constraints. Then, this paper shows that the NoC design is quite flexible and can be reused in server-CPU and AI-processor. The experiment shows that the NoC can support nearly one hundred server-CPU cores' cache coherence service and maintain low access latency. For AI-Processor, this NoC can provide 16TB/s bandwidth and achieve better performance and energy efficiency than state-of-the-art.

REFERENCES

[1] "Amd cdna 2 architecture," 2021. [Online]. Available: https://www.amd.com/system/files/documents/amd-cdna2-white-paper.pdf

[2] F. Alazemi, A. Azizimazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 492–503.

[3] R. Ausavarungnirun, C. Fallin, X. Yu, K. K.-W. Chang, G. Nazario, R. Das, G. H. Loh, and O. Mutlu, "Design and evaluation of hierarchical rings with deflection routing," in *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*. IEEE, 2014, pp. 230–237.

[4] N. Beck, S. White, M. Paraschou, and S. Naffziger, "'zeppelin': An soc for multichip architectures," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 40–42.

[5] A. Biswas, "Sapphire rapids," in *2021 IEEE Hot Chips 33 Symposium (HCS)*. IEEE Computer Society, 2021, pp. 1–22.

[6] C. Chen, Z. Tao, and J. San Miguel, "Bufferless nocs with scheduled deflection routing," in *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2020, pp. 1–6.

[7] M. Evers, L. Barnes, and M. Clark, "Next generation "zen 3" core," in *2021 IEEE Hot Chips 33 Symposium (HCS)*. IEEE, 2021, pp. 1–32.

[8] C. Fallin, C. Craik, and O. Mutlu, "Chipper: A low-complexity bufferless deflection router," in *2011 IEEE 17th International*

Symposium on High Performance Computer Architecture. IEEE, 2011, pp. 144–155.

[9] I. K. Ganusov, M. A. Iyer, N. Cheng, and A. Meisler, "Agilex™ generation of intel® fpgas," in *2020 IEEE Hot Chips 32 Symposium (HCS)*. IEEE Computer Society, 2020, pp. 1–26.

[10] S. Haeri and L. Trajković, "Intelligent deflection routing in buffer-less networks," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 316–327, 2014.

[11] S. Jangam, S. Pal, A. Bajwa, S. Pamarti, P. Gupta, and S. S. Iyer, "Latency, bandwidth and power benefits of the superchips integration scheme," in *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*. IEEE, 2017, pp. 86–94.

[12] M. Li, Q.-A. Zeng, and W.-B. Jone, "Dyxy: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 849–852.

[13] T. Li, J. Hou, J. Yan, R. Liu, H. Yang, and Z. Sun, "Chiplet heterogeneous integration technology—status and challenges," *Electronics*, vol. 9, no. 4, p. 670, 2020.

[14] 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*, 2020.

[15] M. Mattioli, "Rome to milan, amd continues its tour of italy," *IEEE Micro*, vol. 41, no. 4, pp. 78–83, 2021.

[16] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 196–207.

[17] 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)*. IEEE, 2021, pp. 57–70.

[18] 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)*. IEEE, 2020, pp. 44–45.

[19] S. Pal, D. Petrisko, R. Kumar, and P. Gupta, "Design space exploration for chiplet-assembly-based processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 1062–1073, 2020.

[20] I. E. Papazian, "New 3rd gen intel® xeon® scalable processor (codename: Ice lake-sp)." in *Hot Chips Symposium*, 2020, pp. 1–22.

[21] M. Parasar, H. Farrokhbakht, N. E. Jerger, P. V. Gratz, T. Krishna, and J. San Miguel, "Drain: Deadlock removal for arbitrary irregular networks," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 447–460.

[22] M. Parasar, N. E. Jerger, P. V. Gratz, J. S. Miguel, and T. Krishna, "Swap: Synchronized weaving of adjacent packets for network deadlock resolution," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 873–885.

[23] A. Ramrakhyani, P. V. Gratz, and T. Krishna, "Synchronized progress in interconnection networks (spin): A new theory for deadlock freedom," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 699–711.

[24] A. Ramrakhyani and T. Krishna, "Static bubble: A framework for deadlock-free irregular on-chip topologies," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 253–264.

[25] P. Vivet, E. Guthmuller, and e. a. Thonnart, "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, 2020.

[26] Y. Wu, L. Wang, X. Wang, J. Han, S. Yin, S. Wei, and L. Liu, "A deflection-based deadlock recovery framework to achieve high throughput for faulty nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[27] X. Xiang, P. Sigdel, and N.-F. Tzeng, "Bufferless network-on-chips with bridged multiple subnetworks for deflection reduction and energy savings," *IEEE Transactions on Computers*, vol. 69, no. 4, pp. 577–590, 2019.

[28] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. E. Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 726–738.

[29] T. Yoshida, "Fujitsu high performance cpu for the post-k computer," in *Hot Chips*, vol. 30, 2018.