面向通用处理器芯粒架构探索和评估的系统级模拟器

张聪武[①②] 刘 澳[①③] 张 科*[①②] 常轶松[①②] 包云岗[①②]

[①](中国科学院计算技术研究所处理器芯片全国重点实验室 北京 100190)

[②](中国科学院大学计算机科学与技术学院 北京 100049)

[③](郑州大学河南先进技术研究院 郑州 450003)

摘 要：随着摩尔定律的逐步失效，芯片制造工艺的提升愈发困难，芯片性能的提升面临“面积墙”问题，chiplet(芯粒)技术开始被广泛采用来解决此问题。然而，面向chiplet引入的架构设计参数，目前的体系结构模拟器面临新的挑战。为了能够探索chiplet架构的特定设计参数，现有工作通常只会为模拟器增加单一的功能，导致其难以用于探索多个参数对chiplet芯片的整体影响。为了能够较为全面地探索和评估chiplet芯片架构，该文基于现有gem5模拟器实现了面向通用处理器芯粒架构探索和评估的系统级模拟器(SEEChiplet)模拟器框架。首先，总结了现在chiplet芯片设计关注的3类设计参数，包括：(1) 芯片cache系统设计；(2) 封装方式模拟；(3) chiplet间的互连网络。其次，针对上述3类参数：(1)设计并实现了私有末级缓存系统，扩大了cache系统设计空间；(2) 修改了gem5已有的全局目录，以适配私有末级缓存(LLC)系统；(3) 建模了两种常见的chiplet封装方式以及chiplet间互连网络。最后，该文在SEEChiplet框架中进行了系统级的模拟评估，在被测chiplet架构通用处理器上运行操作系统及PARSEC 3.0基准测试程序，验证了SEEChiplet的功能，证明SEEChiplet可以对chiplet设计空间进行探索和评估。

关键词：芯粒；设计空间探索；体系结构模拟器；缓存系统

中图分类号：TN4; TN319 **文献标识码**：A **文章编号**：1009-5896(2024)12-4575-14

DOI: 10.11999/JEIT240299

A System-level Exploration and Evaluation Simulator for chiplet-based CPU

ZHANG Congwu[①②] LIU Ao[①③] ZHANG Ke[①②]

CHANG Yisong[①②] BAO Yungang[①②]

[①](*State Key Lab of Processors*, *Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

[②](*School of Computer Science and Technology*, *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

[③](*Henan Institute of Advanced Technology*, *Zhengzhou University*, *Zhengzhou* 450003, *China*)

Abstract: As Moore's Law comes to an end, it is more and more difficult to improve the chip manufacturing process, and chiplet technology has been widely adopted to improve the chip performance. However, new design parameters introduced into the chiplet architecture pose significant challenges to the computer architecture simulator. To fully support exploration and evaluation of chiplet architecture, System-level Exploration and Evaluation simulator for Chiplet (SEEChiplet), a framework based on gem5 simulator, is developed in this paper. Firstly, three design parameters concerned about chiplet chip design are summarized in this paper, including: (1) chiplet cache system design; (2) Packaging simulation; (3) Interconnection networks between chiplet. Secondly, in view of the above three design parameters, in this paper: (1) a new private last level cache

收稿日期：2024-04-19；改回日期：2024-11-11；网络出版：2024-11-19

*通信作者： 张科 zhangke@ict.ac.cn

基金项目：中国科学院战略性先导科技专项(XDA0320000, XDA0320300)，国家自然科学基金重大项目(62090020)

Foundation Items: The Strategic Priority Research Program of Chinese Academy of Sciences (XDA0320000, XDA0320300), The Major-Program of the National Natural Science Foundation of China (62090020)

system is designed and implemented to expand the cache system design space; (2) existing gem5 global directory is modified to adapt to new private Last Level Cache (LLC) system; (3) two common packaging methods of chiplet and inter-chiplet network are modeled. Finally, a chiplet-based processor is simulated with PARSEC 3.0 benchmark program running on it, which proves that SEEChiplet can explore and evaluate the design space of chiplet.

Key words: Chiplet; Design space exploration; Computer architecture simulator; Cache system

1 引言

近年来，随着机器学习、人工智能等高性能计算等领域的迅速发展，人们对于算力的需求在不断提升。在此背景下，众核通用处理器被广泛用于数据中心。然而，随着摩尔定律(Moore's law)[1]和登纳德缩放(Dennard scaling)[2]的逐渐失效，众核处理器的性能提升面临着愈发严重的“面积墙”问题[3,4]，即通过增大芯片面积来集成更多晶体管的手段面临着技术上的瓶颈：芯片制造工艺节点发展缓慢，导致芯片单位面积的晶体管数量增加变得困难，通过增大芯片整体面积的手段则是面临芯片制造良品率低，制造成本高的问题。

为了解决上述“面积墙”问题，芯粒(chiplet)技术[5–8]开始成为业界关注的重要技术。chiplet技术的核心是将大面积的芯片分解成多个小面积的chiplet，然后使用先进封装技术，如多芯片模块技术(Multi-Chip Module, MCM)、中介层(interposer)等[5,6,8]，将这些chiplet集成在裸片(die)上并组合到一起，从而构建大面积的芯片。使用小面积的chiplet能够有效解决大面积芯片制造良品率低的问题；且基于chiplet架构的系统不受芯片面积限制，通过集成多个chiplet来集成更多晶体管到芯片上。chiplet架构逐渐成为现有芯片设计的主流选择，如超威半导体(Advanced Micro Devices, AMD)的Zen系列通用处理器[9–12]，Intel最新的Xeon处理器[13]和Saphire FPGA[14]，基于开源指令集的大芯片(big chip)[3]、Veyron[15]等。

将整体芯片分解为多个chiplet并重新组合的方式为系统设计带来了更多的设计选择。如表1所示，chiplet本身的设计选项以及chiplet间的互连架构(interconnect fabric)均使得chiplet架构的设计空间变大。由于chiplet架构的性能主要受到chiplet间数据交互的影响，目前对于chiplet架构的设计探索主要集中于chiplet缓存系统设计[16–21]以及chiplet间的拓扑网络[22,23]。为了快速探索chiplet架构的设计空间，研究者在完成硬件设计前，需使用体系结构模拟器对chiplet架构的设计空间进行系统级架构探索及性能评估，确保设计参数使得chiplet架构能达到预期性能指标要求。然而，现有的模拟器在探索和评估chiplet架构时，面临着以下两个问题：

(1)现有工作探索的设计空间有限。目前，研究chiplet的大部分工作在探索相同设计空间不同设计参数下chiplet架构的性能表现，而非提供探索chiplet架构设计空间的工具。例如，文献[21]探索了采用不同末级缓存(Last Level Cache, LLC)组织的chiplet架构性能变化，文献[23]探索了人造流量在不同chiplet拓扑下的chiplet交互情况。这些工作仅建模了某一个设计参数，并没有对整体的chiplet架构设计空间进行探索。

(2)模拟器种类繁多，难以横向比较。在研究chiplet时，研究人员根据需求采用了不同的软件模拟器，如PriME[24], gem5[25]等。由于不同模拟器具有不同的模拟方法和目标，例如gem5使用了事件驱动的方法来构建周期级的全系统模型，PriME则是利用功能模拟的方法构建了众核系统模型。这种差别导致难以将基于不同模拟器的chiplet工作综合到一起。另外，现有的模拟器都是针对传统系统架构进行模拟的工具，需要重新建模chiplet新引入的特性，如chiplet封装，chiplet间的拓扑网络等。

针对上述问题，本文基于gem5模拟器实现了面向通用处理器芯粒架构的系统级探索评估模拟器(System-level Exploration and Evaluation simulator for Chiplet, SEEChiplet)模拟器框架。本框架集成了目前研究者关注的chiplet设计参数，能够

表 1 众核chiplet架构设计空间

设计选项	参数数量
chiplet本身	处理器：指令集架构；顺序执行，乱序执行；核心数量
	cache系统：cache 块大小；cache容量；cache层级；chiplet私有末级缓存，全局共享末级缓存等
	chiplet数量
chiplet互连架构	chiplet拓扑：Mesh, IO-die等；路由算法
	chiplet互连：连接带宽；连接延迟；Router延迟
	chiplet集成方式：MCM, 2.5D, 3D；chiplet与封装基板或中介层(Interposer)间SERDES配置等

模拟在真实系统中探索chiplet架构设计空间，对其进行系统层面的评估。同时，本框架丰富了面向chiplet架构的数据统计，能够提供更多针对chiplet架构的评估内容。

本文的贡献：

(1) 本文设计并实现了面向chiplet的私有末级缓存架构，提供了多种chiplet cache系统设计空间参数，允许对chiplet cache系统进行探索和评估；

(2) 本文基于gem5的全局目录进行了面向chiplet架构的私有末级缓存系统的适配，重新对全局目录建模，实现了基于目录的cache一致性协议；

(3) SEEChiplet建模了两种chiplet封装方式以及chiplet间网络，提供了网络连接参数与拓扑的探索和评估。

2　技术背景及研究动机

2.1　众核chiplet架构研究

如图1所示，在引入chiplet技术后，众核系统在系统结构上发生了变化：传统架构中，芯片上集成多个处理器核心和cache系统；在chiplet架构中，处理器核心和cache系统分布在多个裸片上，每个裸片为一个chiplet, chiplet通过先进封装技术集成到封装基板上。在chiplet架构中，可以按照芯片功能模块解耦，如计算chiplet、输入输出chiplet等。使用这种解耦方式，可以简化chiplet的设计，同时，计算chiplet上可以留出更多的晶体管给到cache系统，增加芯片的cache容量。这点在现有基于chiplet的芯片中有所体现，如AMD的Zen4[12]架构，计算chiplet上可以部署32MB的末级缓存。然而，相比之前在同一个芯片上的cache系统，分布在不同chiplet上的cache系统带来的新的设计选择。

chiplet之间的连接性能，如带宽、延迟等，会受到封装技术的影响[16]。因此，为了获取更详细的评估，模拟器需建模chiplet封装方式。此外，还需建模chiplet架构新引入的chiplet间互连网络。

2.1.1　众核chiplet cache系统

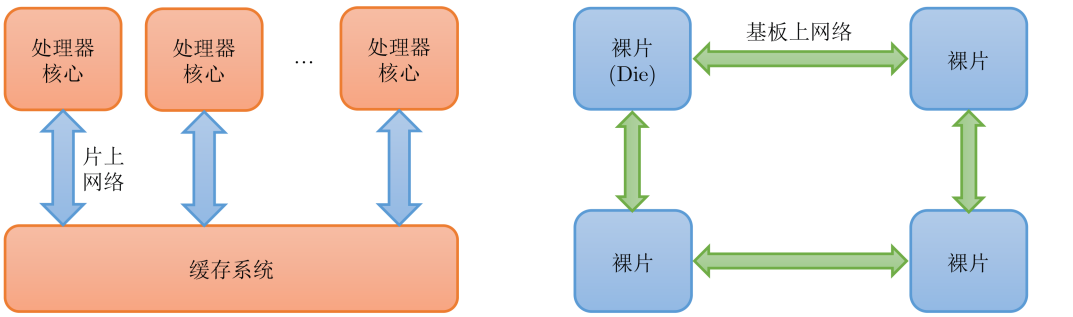
随着处理器与内存速度差异的扩大，现代处理器引入了cache系统，通过将数据缓存在更接近处理器的位置来降低速度差异带来的影响。然而，cache中的数据可能与内存不一致，需要cache一致性协议以确保cache和内存间的数据一致性。常用的cache协议是MESI(Modified, Exclusive, Shared or Invalid)及其变种，通过一致性信息同步cache块的状态。cache之间的信息同步方式分为两种：(1) 基于广播的同步机制，即将同步信息广播给所有cache控制器。随着核数增加，cache同步消息也呈指数级增长[20]；(2) 基于目录的信息同步机制。处理请求时根据目录中记载的cache块的信息，将同步信息发给特定的cache控制器上，减少了不必要的同步信息传输。

在众核chiplet芯片中，cache间的同步信息通过片间网络交互，这些信息主要来自末级缓存，因此chiplet芯片末级缓存的设计备受关注。目前，众核chiplet芯片末级缓存有两种主流的设计：(1) 如图2(a)所示，末级缓存被所有的chiplet上的核心共享，我们称之为全局共享末级缓存(Shared LLC)。此末级缓存由多个bank组成，每个bank映射不同内存地址。如图2(a)所示，两个bank映射的地址范围互不包含。(2) 如图2(b)所示，每个chiplet都有私有的末级缓存(Private LLC)，该末级缓存映射全部的内存地址空间。这种末级缓存架构能够充分利用随着chiplet的技术发展而带来的cache容量增加的好处[21]。

2.1.2　chiplet的集成与互连

chiplet封装方式的选择会影响chiplet和封装基板之间的连接性能以及chiplet之间的互连性能，不同的集成方式需要的成本也不同，因此在设计chiplet架构时，需要考虑选择合适的封装技术。

现在主流的chiplet封装技术包括了两种，第1种是MCM封装，也被称为2D封装。这种封装方式



(a) 传统芯片结构　　(b) 众核chiplet架构, 裸片集成多个处理器核心和缓存系统

图 1 传统芯片结构和众核chiplet结构对比

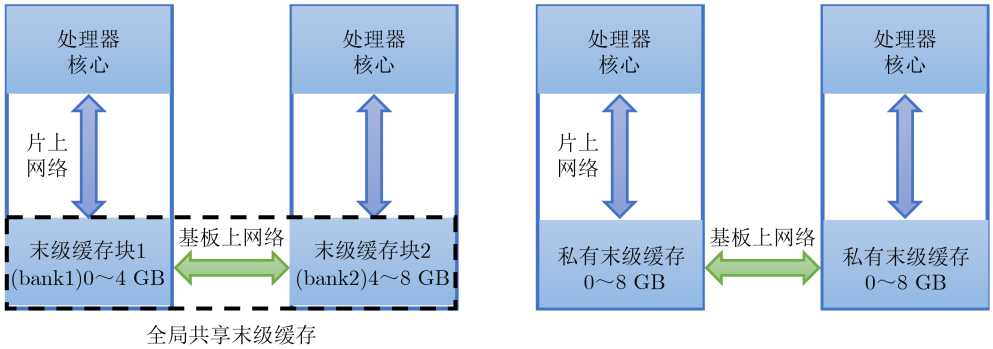
如图3(a)所示，将chiplet放在封装基板上，chiplet间通过封装基板内部电路通信。第2种是基于硅中介层(interposer)的互连，chiplet贴装在中介层(Interposer)之上，如图3(b)所示，相比MCM来说有更高的带宽和更低的延迟，不过成本会更高。第2种方式相比2D封装多了一层硅中介层，因此也被称为2.5D封装。除了上述两种封装方式以外，基于3D封装的chiplet技术也在不断发展，目前这种技术主要应用在存储方面，本文暂不考虑实现。

2.2 chiplet架构模拟工具

目前chiplet研究主要使用软件模拟器对chiplet架构进行模拟，研究chiplet之间的交互。表2中列出了近些年chiplet研究的相关工作。其中大部分工作是在研究cache系统和chiplet拓扑对系统的影响，这部分工作在利用现有体系结构模拟器探索某一设计参数对chiplet架构的影响。另外一部分工作探索了如何模拟chiplet架构：Zhi等人[32]基于gem5[24]和BookSim[29]搭建了多chiplet模拟器，此工作在多个gem5实例上运行了相同程序并使用BookSim记录程序间交互，获取多chiplet加速器的数据交互表现。该工作中每个gem5实例独立运行操作系统，接近于集群模拟，与真实chiplet架构差距较大。muchiSim[34]构建了众核chiplet加速器模型，并对这些模型间的交互进行详细建模。上述工作建模了面向特定应用的chiplet架构软硬件协同优化，并不适合用于其他情况。文献[35]利用FPGA构建chiplet系统集成模拟，但并没有实际测试。SMAPPIC[36]基于FPGA实现了多chiplet众核仿真框架，然而该工作并没有对chiplet集成方式建模，且chiplet间网络参数不可配置，另外该工作仅支持全局共享末级缓存形式的缓存系统，并不支持chiplet私有末级缓存。表3总结了这些相关工作和本文工作的特性，说明上述工作对于chiplet架构设计空间的模拟有限，难以用于探索chiplet架构的设计空间。

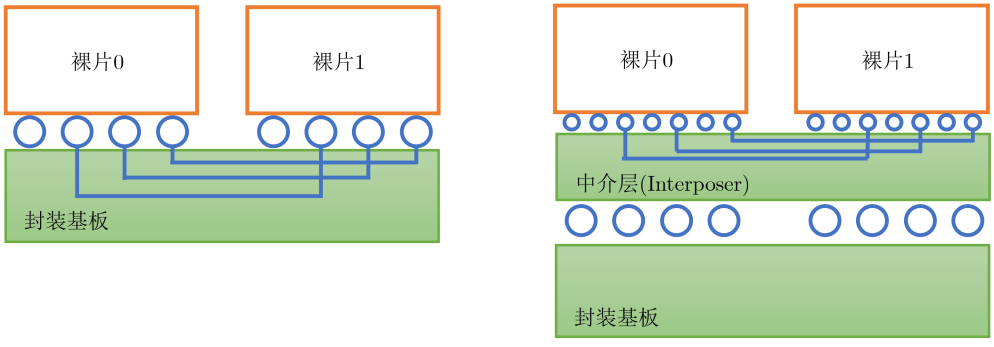
2.3 gem5模拟器

目前，在chiplet架构的研究工作中，有不少工作是基于gem5模拟器的。gem5是一款开源计算机体系结构研究模拟器，采用模块化的建模方法模拟计算机体系结构的各个组件，如CPU，片上网络，内存等，被广泛应用于计算机系统架构和处理器设计研究领域。然而，gem5缺乏了对chiplet架构的针对性模拟。例如，对上一节提到的众核chiplet芯片cache系统中，gem5目前仅支持全局共享末级缓存的模拟，并不支持chiplet私有末级缓存的模拟。另外，gem5缺少了对chiplet架构的封装和基板上网络的建模。



(a) 共享末级缓存 chiplet cache 系统　　(b) 私有末级缓存 chiplet cache 系统

图 2 两种众核chiplet cache系统(以内存容量8GB为例)



(a) MCM（2D）封装　　(b) 2.5D 封装

图 3 chiplet集成方式

表 2　现有chiplet研究工作

研究工作	基于的模拟器或模拟手段	研究内容	chiplet封装方式	是否支持模拟运行操作系统	是否开源
Meduza[16]	PriME[24]	chiplet cache系统	2.5D	否	否
文献[17]	gem5[25]	chiplet cache系统	2.5D	否	否
文献[18]	Multi2Sim[26]	chiplet cache系统	无线连接	否	否
文献[19]	gem5-X[27]	chiplet cache 系统	无线连接	是	否
1-Update[20]	SimFlex[28]	chiplet cache系统	3D	否	是
SILO[21]	未提到	chiplet cache系统	3D	否	否
Kite[22]	gem5	chiplet 拓扑	2D, 2.5D	是	否
HexaMesh[23]	BookSim[29]	chiplet 拓扑	2.5D	否	否
文献[30]	Swarm[31]	chiplet 架构性能	2D, 2.5D	否	否
文献[32]	gem5[24]	chiplet 架构模拟	无	是	是
DCRA[33]	muchiSim[34]	chiplet 架构模拟	2D, 2.5D	否	是
文献[35]	FPGA	chiplet 架构模拟	2D	是	否
SMAPPIC[36]	FPGA	chiplet 架构模拟	无	是	是

表 3　chiplet模拟器相关工作比较

chiplet模拟工作	全系统模拟	chiplet集成	cache系统	chiplet间网络
文献[32]	×	×	×	√
muchiSim[34]	×	×	×	√
文献[35]	×	√	×	×
SMAPPIC[36]	√	×	√	×
SIAM[37]	×	×	×	√
SEEChiplet	√	√	√	√

gem5的核心设计是一个事件驱动引擎(event-driven engine)，通过真实系统的所有硬件行为看作不同事件，并通过维护全局事件队列，将事件插入到队列中并按周期顺序执行，从而构建了周期级精确的模型。以gem5的片上网络模型为例，如图4所示，发送方发送消息的行为被看作一个事件，并在第k个周期执行该事件，执行时会根据连接方向判断接收方是谁，在事件队列的第$k+n$个周期处插入接收方的接收事件，从而在功能和时序上模拟真实系统中的发送接收行为，保证模拟的结果和真实系统一致。

3　SEEChiplet框架

本节首先介绍了SEEChiplet整体框架，包括为了支持对chiplet架构的探索和评估所做的改变，接着详细介绍了SEEChiplet框架中引入的chiplet私有末级缓存组织架构实现和chiplet封装的实现和在这些改进中引入的新的评估内容。

3.1　SEEChiplet介绍

如图5所示，SEEChiplet基于gem5多核模拟功能，新增了针对chiplet架构的模拟，提供了更加丰富的chiplet架构设计空间参数。相比之前的模拟

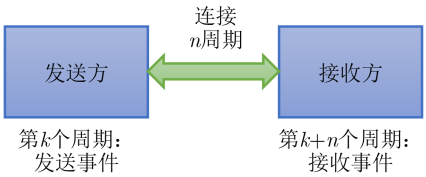


图 4 gem5片上网络连接模型

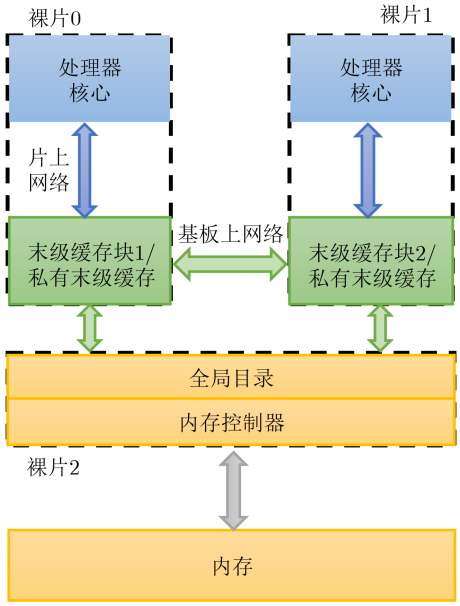


图 5 SEEChiplet整体框架

工具，SEEChiplet全面结合了chiplet私有末级缓存的组织方式，片上网络(Network-on-Chip, NoC)和基板上网络(Network-on-Interposer, NoI)的模拟，两种chiplet封装方式等针对chiplet架构的设计参数，并且利用了gem5本身的全系统模拟以及周期级模拟能力，支持对chiplet架构进行全面且准确的建模。

chiplet私有末级缓存系统建模。如图6(a)所示为私有末级缓存的建模。首先为了能够维护多个末级缓存的cache块，SEEChiplet采取了全局目录的方式来记录cache块的状态以及归属。相比广播机制来说，基于目录的cache一致性协议只需要访问记录的chiplet，从而减少了chiplet间的数据交互。其次，私有末级缓存的行为需要重新建模：(1)如图6(a)中编号中①所示为chiplet内部的数据交互。因为私有末级缓存持有整体内存地址空间的数据，这和之前的全局共享末级缓存的bank不同，后者只能持有部分内存地址空间的数据，因此私有末级缓存一定会处理来自2级缓存的请求并响应；(2)图6(a)中编号②所示为2级缓存与末级缓存。因为目录记录了cache块的持有者，因此需要将对应地址的请求转发给持有者，并且等待持有者的响应；(3)图6(a)中编号③所示为末级缓存之间数据交互行为。由于其他末级缓存完成请求之后需要通知发出请求的chiplet，因此增加了末级缓存之间的数据通路，缩短请求返回到目录并由目录返回到请求chiplet的过程。

chiplet互连架构建模。chiplet互连架构的建模包含了两个部分：(1)chiplet集成方式的建模。使用不同的封装方式时，chiplet与封装基板间以及chiplet之间的连接带宽与延迟有所不同；(2)chiplet间互连网络建模。SEEChiplet建模了chiplet间网络，并提供了网络连接参数的配置以及两种拓扑。

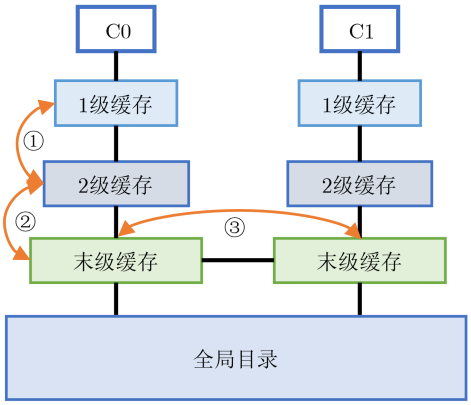
SEEChiplet建模准确度。SEEChiplet基于gem5的周期级cache模型建模了私有末级缓存系统和chiplet间互连架构。该建模中的事件按照周期顺序执行，确保其能在周期上对齐真实系统的行为。

3.2 设计并实现私有末级缓存

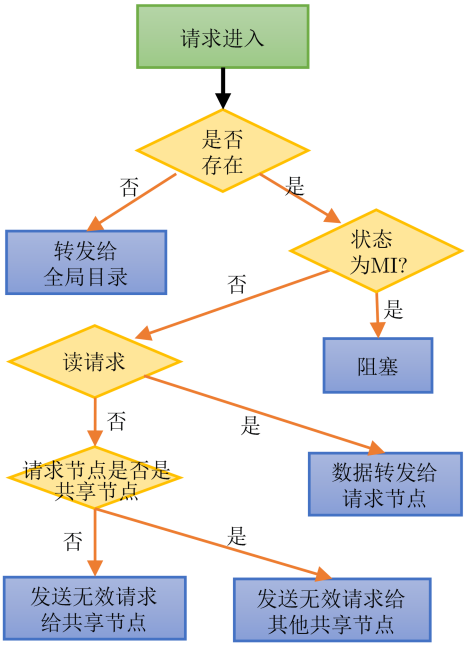
SEEChiplet在gem5的MESI协议基础上，开发了一种面向chiplet架构的私有末级缓存的cache组织方式。本文首先设计了一种阻塞确认机制，同时修改了cache读写请求处逻辑。另外，本文引入了一个特殊的共享状态表示cache块仅在当前chiplet，从而减少不必要的chiplet间交互：

请求阻塞确认机制。请求阻塞发生在当前chiplet想要独占某个cache块时，若当前cache块存在且并未阻塞，则将此cache块设为中间(Medium, MI)阻塞状态。请求确认包含两个阶段：第1个阶段等待来自其他chiplet的确认信息：若存在其他chiplet持有相同地址的cache块，则需无效化并通知当前chiplet，从而获取当前cache块的独占权；第2阶段是当前chiplet在获取独占权之后，需发送确认信息给全局目录，全局目录修改cache块的状态为独占。

私有末级缓存请求处理。chiplet私有末级缓存中的cache块可能会被其他chiplet持有，因此末级缓存的读写请求，需要其他chiplet参与。面对来自上一层级的读写请求，如果请求的数据被当前chiplet独占，则直接在此末级缓存完成请求；如果在内存或者其他chiplet中存在，那么就需要将请求转发给全局目录，全局目录根据具体情况处理。图6(b)是简要的末级缓存请求流程，对于写请求，



(a) 私有末级缓存整体建模



(b) 私有末级缓存读写请求处理流程

图 6 chiplet私有末级缓存系统建模以及读写请求处理流程

考虑到在其他chiplet中的备份，需要等待全部无效确认之后，才可以完成请求。

特殊的共享状态。对于cache块的共享状态，SEEChiplet还增加了一个chiplet内部共享(Private Shared, PS)的状态，这个状态表明cache块被当前chiplet独占，由chiplet中的多个核心共享。设置这个状态可以细分处于共享状态的cache块处理的情况，减少不必要的chiplet间的流量。如面对写请求，只需要在chiplet内部执行即可，不需要考虑其他chiplet持有cache块的情况。

3.3 SEEChiplet全局目录实现

cache同步机制主要分为两类：广播机制以及目录。由于广播机制会导致chiplet间交互随着chiplet数量的增多而指数级增长[21]，因此SEEChiplet采用了基于目录的cache架构。

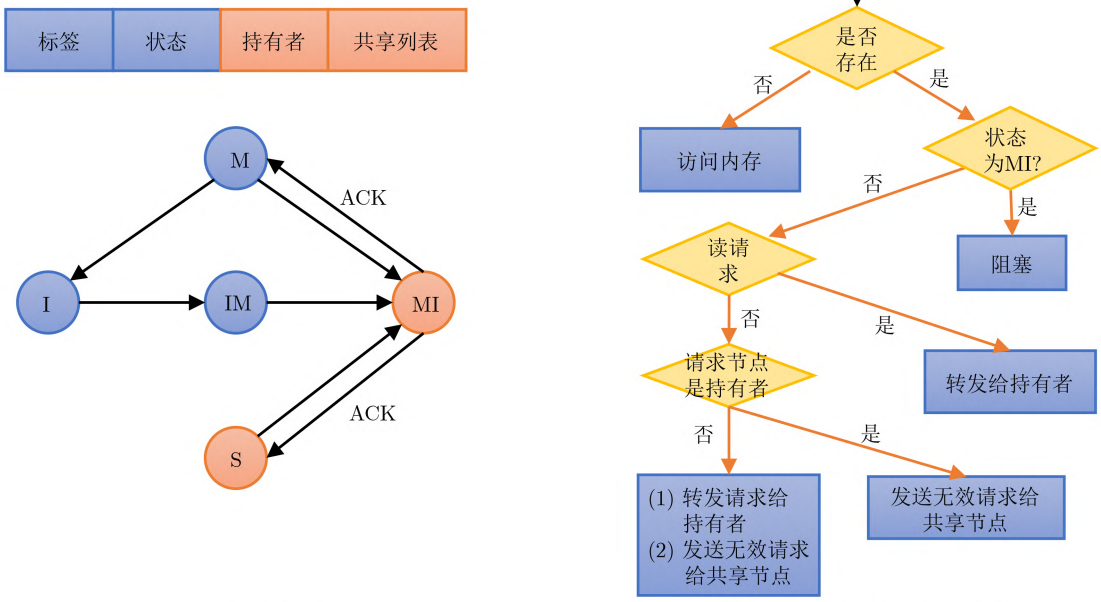
缓存目录设计。基于目录的Private LLC实现过程中，首先需要考虑目录的实现方式。原有的全局目录用于同步全局共享末级缓存，并没有考虑在末级缓存上存在多个相同地址的数据的情况。因此，SEEChiplet针对私有末级缓存这一架构，扩展了全局目录，此目录作为单独的chiplet存在于chiplet拓扑中，用于维护chiplet私有末级缓存的一致性。

图7(a)展示了全局目录表项的组织结构以及全局目录的简要状态机模型，其中蓝色部分为gem5全局目录原有的字段，橙色为新增的字段。在原有gem5全局目录中，只记录了当前cache块的标签和状态。然而chiplet私有末级缓存架构中，不同chiplet的末级缓存是独立的，会出现同时持有相同cache块的状态。在处理请求时，需要获知当前cache块的持有者，请求转发到对应的chiplet，获取一致的cache数据。对于处于共享状态的cache块，需要记录共享节点列表，方便在特定请求发生时对这些共享节点同时处理，避免出现不一致的情况。

请求阻塞确认机制。在chiplet架构中，全局目录需要同时处理多个chiplet的请求，而且不同chiplet到全局目录的距离也不尽相同。为了能够保证cache系统的一致性，SEEChiplet的全局目录引入了请求阻塞确认机制：处理请求时，全局目录首先将对应地址的表项设置为阻塞状态MI，缓存其他chiplet对此地址的请求，保证当前请求完成之后再进行其他请求的处理。如图7(b)所示，在收到请求节点的确认信息以后，状态转为稳定，唤醒其他请求队列中相同地址的请求。

读写请求处理。全局目录负责记录当前cache块在chiplet节点中的状态，也负责请求的转发。请求处理逻辑如图7(b)所示，当请求到来时，首先判断cache块是否在chiplet架构中存在，如果存在，那么就将该请求转发给Owner，用来获取最新的数据；否则就将请求转发给内存控制器来从内存中获取对应的数据。

实现广播机制。文献[20]使用广播机制来进行



(a) 全局目录建模　　(b) 全局目录读写请求处理流程

图 7 全局目录建模及读写请求处理流程示意

cache同步，因此在chiplet架构中广播机制仍有存在的必要。SEEChiplet可经过简单扩展实现广播机制。如图6(a)所示，从末级缓存出来的请求经过全局目录筛选后转发给存在目标数据的chiplet节点，如果略去这步，直接通过路径,将请求转发给其他所有的chiplet节点，每个chiplet节点在收到请求后对其进行筛选，存在相应数据即进行相应处理，即可实现广播机制。

3.4 chiplet互连

chiplet封装建模。设计chiplet架构时，不同的封装方式的连接参数也不同。因此，在模拟时需要考虑对不同封装进行建模。SEEChiplet中MCM封装和2.5D封装建模方案见图8。对于MCM封装建模，每个chiplet内部有一个router，负责chiplet和外部的消息交互，末级缓存发出的请求通过此router和directory或其他末级缓存进行交互。对于2.5D封装，SEEChiplet建模了interposer上的router，该router负责和chiplet上的router互连以及根据chiplet的拓扑进行消息的转发。除此之外，对于2.5D封装的chiplet架构，还需要建模chiplet和interposer之间连接的串并转换(SERializer/DESerializer, SERDES)和跨时钟转换(Clock Domain Crossing, CDC)，实现带宽转换和时钟转换的功能。

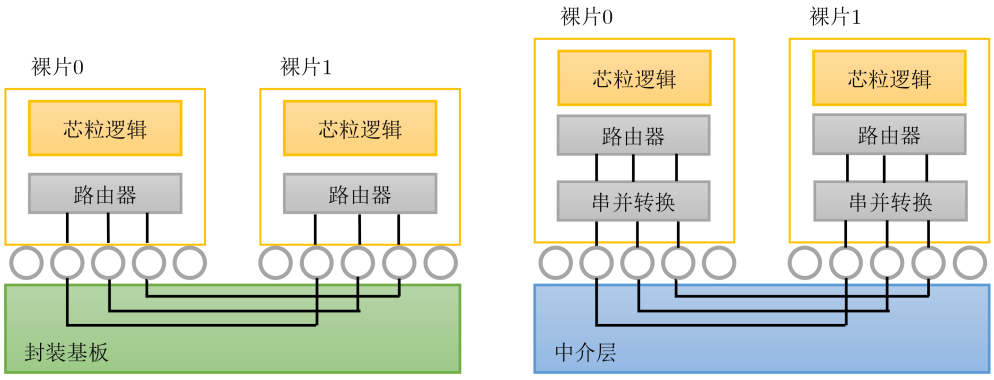
chiplet互连网络建模。chiplet间互连网络的建模包括两部分：(1) 网络连接参数建模；(2) 拓扑建模。gem5中已有的HeteroGarnet[22]模块可以用于建模片上互连网络。然而，片上网络的配置和chiplet间网络的配置不同。SEEChiplet基于HeteroGarnet重新建模了chiplet间互连网络，提供了chiplet之间的互连进行配置参数。此外，SEEChiplet还支持了两种chiplet拓扑，可以探索拓扑的设计空间。

4 实验参数及环境

4.1 模拟器参数配置

表4为实验时SEEChiplet支持的参数配置，同时SEEChiplet继承了gem5的全系统模拟功能，可以启动操作系统，允许对chiplet架构进行系统级的探索和评估。实验时，本文选择了在模拟的操作系统中运行PARSEC 3.0 benchmark[37]，来统计运行工作负载时的架构相关数据，该数据可用于分析chiplet架构的相关性能指标。

实际运行测试时，SEEChiplet首先使用基于内核的虚拟机(Kernel-based Virtual Machine, KVM)来快速启动操作系统，KVM利用现代处理器的硬件虚拟化技术使得SEEChiplet快速跳过操作系统启动流程，启动完成之后，切换到模拟的待测



(a) MCM封装 (b) 2.5D封装

图 8 不同封装方式的实现方案

表 4 SEEChiplet模拟参数配置表

配置项	基本信息
CPU	Timing CPU, X86指令集，3 GHz
cache层级及相应参数(其中容量等参数可以根据用户需求配置)	3级cache, Inclusive，频率同CPU L1: 指令cache，数据cache；每个CPU核心一组；均为32 kB, 4路组相连 L2: 每个CPU一组；1MB, 8路组相连 L3：所有chiplet共享/chiplet内部共享；32 MB, 16路组相连
封装方式	MCM, 2.5D：SERDES组件增加2个cycle, Router本身3个cycle
chiplet拓扑	支持IO Die, Mesh架构
chiplet参数	每个chiplet可以有2, 4, 8, 16个核心
内存	单通道DDR4, 8 GB, 2400 MT/s

chiplet处理器，运行并收集待测程序实际运行过程中产生的数据，包括CPU运行相关的数据，cache中发生的状态转化，数据包在chiplet内部和chiplet之间的交互信息等。

目前SEEChiplet支持两种拓扑IO Die和Mesh两种拓扑分别对应图9(a)、图9(b)。此外，SEEChiplet还支持不同的封装方式和不同的LLC组织方式。两种封装方式分别被称为MCM和2.5D。两种LLC组织方式分别为dbank(distributed bank)和private(chiplet-private LLC)，前者为全局共享末级缓存，将末级缓存的bank分布到不同的chiplet上，后者则是每个chiplet独占末级缓存。

5　实验结果分析

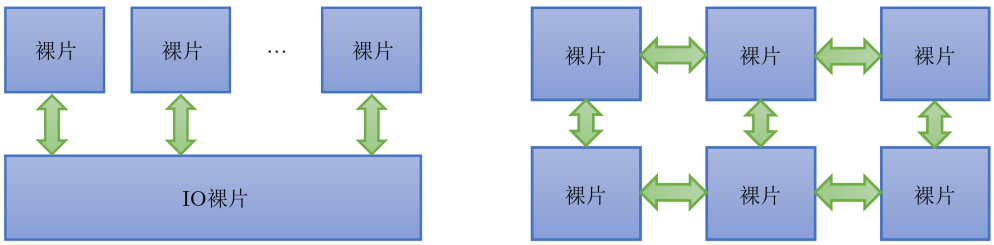
实验主要目的是对SEEChiplet的功能进行分析，通过运行多个测试程序，收集测试程序的ROI运行时间和相关数据，来证明SEEChiplet能够对不同chiplet设计空间的探索。实验首先通过在SEEChiplet上运行多种不同的程序来验证SEEChiplet可以模拟全系统环境下的多个程序。之后，通过设置不同的设计空间参数，构建不同的chiplet架构结构，在这些系统结构下运行相同的程序，并通过运行结果来分析不同chiplet设计空间下的表现。本文主要关注了不同chiplet cache系统的选择以及不同chiplet封装方式和网络拓扑对chiplet架构性能的影响。

5.1　功能分析

本次实验主要目的是对SEEChiplet的功能进行分析，通过运行多个测试程序，收集测试程序的ROI运行时间和相关数据，来验证SEEChiplet对不同类型的测试的支持。本次实验固定了基本的模拟环境配置：其中，CPU核数为32，chiplet数目为16，即每个chiplet中有两个CPU核心；分别测试了2D和2.5D两种封装方式，在mesh拓扑下的表现。图10展示了本次实验结果，每个子图横坐标为不同的测试程序代号，纵坐标为程序在模拟环境中运行的时间。

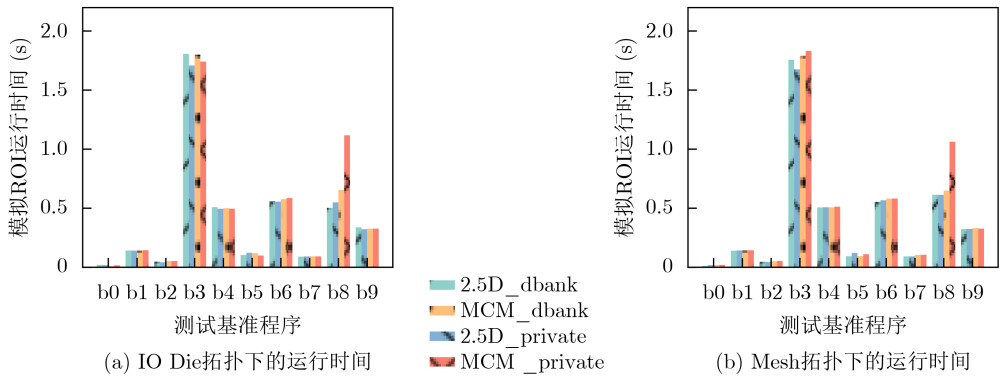
5.2　cache组织结构对比分析

本次实验主要目的是对SEEChiplet新增的私有末级缓存与全局共享LLC的cache组织架构进行对比测试。实验主要通过运行blackscholes程序，收集了待测程序运行时每个包从发出到接收的平均延迟以及末级缓存请求分布，对两种组织结构的形式进行对比。本次实验首先收集了在不同核数、不同chiplet数以及不同拓扑下，MCM和2.5D封装的程序运行结果，结果见图11。每张子图代表了不同的chiplet数目下不同核数的测试结果，每张子图横坐标为处理器核心数目，纵坐标为程序运行过程中，数据包从发出到被接收的平均延迟。私有末级缓存的平均包延迟比共享缓存的平均包延迟低了37.7%，该降低主要来源是由于私有末级缓存架构中，数据包大部分会在chiplet内部被处理，使得平



(a) IO-Die chiplet 拓扑　　(b) Mesh chiplet 拓扑

图 9 IO-Die和Mesh拓扑



(a) IO Die拓扑下的运行时间　　(b) Mesh拓扑下的运行时间

图 10 模拟环境下不同拓扑参数下基准测试程序运行时间

均包延迟降低。上述结论可以从表5中得出，该表记录了16个chiplet不同末级缓存架构下chiplet内部处理的请求以及需要转发到其他chiplet的外部请求，可以看到私有末级缓存有78.4%的内部请求，而全局共享末级缓存仅有5.8%的请求，这也符合全局末级缓存平均分布到每个chiplet的设计。

5.3 封装方式和拓扑分析

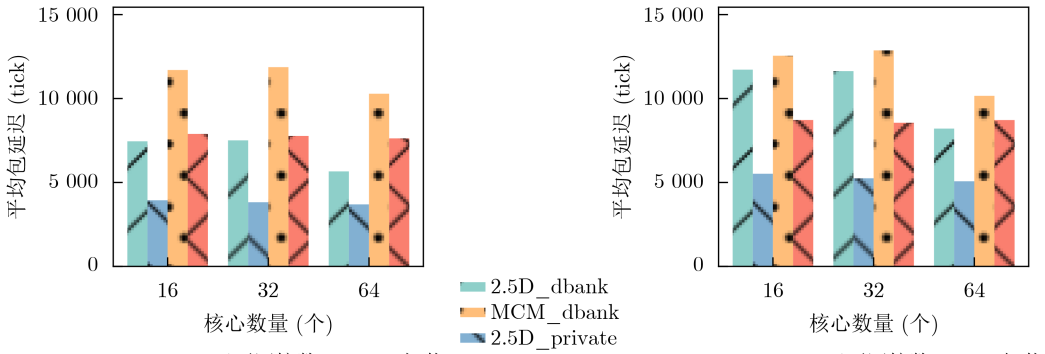
从图11中可以看出，相同架构下，不同封装方式的平均包延迟有明显的差别，即MCM封装相比2.5D 封装来说有更高的延迟，2.5D封装的延迟相比MCM封装延迟减少了23%，这是由于MCM本身的技术特点导致的传输延迟较高。从图12可以看出MCM的平均包跳数是要低于同配置下的2.5D封装，MCM封装平均包跳数比2.5D封装减少了24.8%，但是由于MCM封装技术中裸片通过封装基板进行连接，而2.5D封装裸片通过中介层进行连接，前者具有更高的连接延迟，导致平均包延迟较高。

不同拓扑下，相同处理器数目和chiplet数目的模拟环境中，可以从图11看出mesh拓扑下包的延迟会相对高一些。这是因为使用mesh拓扑下，每个包需要经过多跳，才能到达目的chiplet上，因此平均包延迟会比较高一些。另外，从图12可以看到，不同架构下，IO Die拓扑的平均跳数没有什么变化，mesh拓扑下平均包跳数随着chiplet数目的提升而增加。相同架构下，IO Die拓扑的平均包跳数会低于mesh拓扑，随着chiplet数目的提升，这个现象更加明显。说明IO Die拓扑在包延迟和跳数有优势，但是这种拓扑扩展性不高，难以扩展到更多chiplet的场景。

5.4 SEEChiplet开销分析

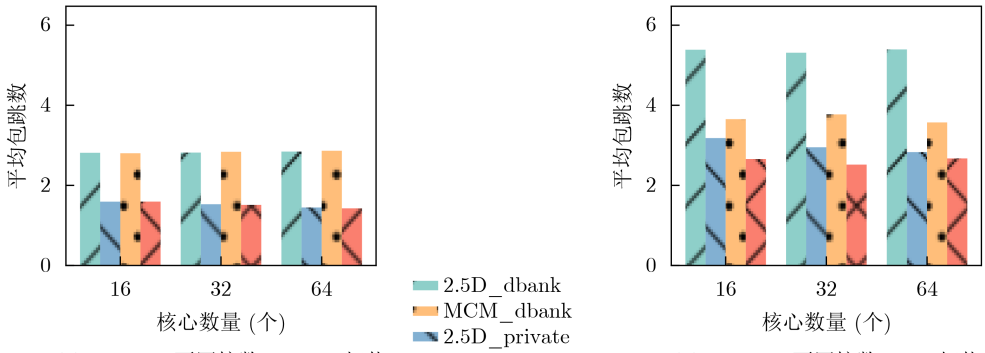
SEEChiplet开销主要分为建模开销以及运行时开销，表6总结了建模私有末级缓存以及建模芯粒间网络带来的开销。可以看到开销主要在私有末级缓存增加的状态以及处理逻辑，全局目录新增的状态以及共享列表以及chiplet间网络本身的开销。

对于运行时开销，我们发现私有末级缓存收到



(a) 16chiplet下不同核数，IO Die拓扑　(b) 16chiplet下不同核数，Mesh拓扑

图 11 16chiplet不同核数不同拓扑，blackscholes程序运行的平均包延迟



(a) 16chiplet不同核数，IO Die拓扑　(b) 16chiplet不同核数, mesh拓扑

图 12 16chiplet不同核数配置下，blackscholes基准测试程序在模拟环境运行时的平均包跳数

表 5 不同末级缓存架构，chiplet内外部请求分布

末级缓存组织形式	内部请求比例(%)	外部请求比例(%)	请求总数量
chiplet私有末级缓存	78.4	21.6	458037
全局共享末级缓存	5.8	94.2	404340

表 6 SEEChiplet建模开销总结

开销来源	开销总结
chiplet私有末级缓存	代码量：～1000行 新增中间状态：12个 新增事件类型：11个 新增状态转移逻辑：30个，和全局目录及其他LLC进行交互 新增虚通道：2个，用于和全局目录进行交互
全局目录	代码量：～600行 每行新增bit数：64bit用于存放共享chiplet列表，8 bit用于存放持有者chiplet ID 新增状态：1个基础状态S, 9个中间状态 修改状态：M状态以及相关处理逻辑 新增事件类型：10个 新增状态转移逻辑：18个，全局目录转发请求，响应请求等 新增虚通道：2个，用于和末级缓存进行交互

包的总数比全局共享末级缓存系统多，具体结果可见表5，经过分析，我们发现额外包的来源主要有两种：(1)来自全局目录的转发。当前私有末级缓存没有对应请求的cache块，在查询全局目录时，发现在此块在其他chiplet上，全局目录会转发请求到持有cache块的chiplet中，该chiplet在处理完成之后也会发送响应包到请求chiplet。因此，增加了chiplet上的请求数量；(2)来自全局目录的无效化请求。当私有末级缓存处于共享状态的cache块遇到写请求时，会发送请求到全局目录去将其他持有者无效化，因此会造成chiplet上的请求增多。

6　相关工作

通过调整末级缓存架构增强缓存访问效率。为了提升访问带宽，文献[38]将整块末级缓存分为多个bank，每个bank映射一部分内存空间，这种方式被称为静态非一致缓存访问(Static Non-Uniform Cache Access, S-NUCA)，gem5的全局共享缓存就是按照S-NUCA进行建模。这种方式会造成访问其他bank的延迟较高。之后，研究人员提出了动态非一致缓存访问(Dynamic Non-Uniform Cache Access, D-NUCA)的架构[39,40]，末级缓存每个bank可以映射所有的地址空间，从而可以减少远程访问的数量。这种方式的缺点在于bank容量较小，数据替换造成的开销较大。随着chiplet技术的采用，每个chiplet可以容纳更多的cache容量，且chiplet间的访问延迟更高，因此D-NUCA被chiplet架构重新采用。SEEChiplet采用这种方式，建模了chiplet私有末级缓存架构。

片上网络系统解决死锁问题。在片上网络解决死锁的方式主要为采用多个优先级不同的通道，来发送请求和接收响应。SMAPPIC[36]构建了3个物理片上网络通道，通道1用于传输请求，通道2用于响应请求，优先级最高的通道3一般用于响应来自其他芯粒的请求或者是写回内存的操作。然而，建模物理通道的方式不容易扩展，通道数增加会带来较大的开销。gem5[25]采用了虚拟通道技术，通过使用多个buffer来复用一条物理通道。SEEChiplet基于gem5的虚通道技术，在原有的全局共享缓存中通过新增两个虚通道，建模了chiplet私有末级缓存系统。

建模chiplet互连架构。chiplet互连架构具有多种连接方式，经常采用的方式为2D, 2.5D封装，模拟器通过使用不同延迟来建模不同的封装方式。除此之外，还有建模基于硅光连接架构的chiplet系统，如文献[41]，通过建模硅光连接高带宽，低延迟的波分复用(Wavelength Division Multiplexing, WDM)通道，来探索硅光连接在chiplet架构中的可行性。另外，还有文献[18]探索基于无线连接在chiplet架构中的可行性。利用无线网络的广播特性，取代了维护cache块信息的全局目录，从而减少了全局目录带来的芯片面积开销。然而，上述两种技术仍在不断发展中，难以取代现有的芯片封装技术。

7　结束语

随着chiplet技术的不断发展，chiplet架构的设计空间随之不断增长。使用软件模拟器探索设计空间，分析不同设计参数下chiplet架构的表现，成为降低硬件设计验证成本的有效手段之一。本文针对现有模拟器有限支持chiplet架构的问题，基于gem5模拟器开发了SEEChiplet框架。该框架支持了众核chiplet架构中常见的cache系统设计、chiplet封装方式以及chiplet间互连网络的设计参数，能够探索这些参数组成的chiplet架构设计空间。本文使用SEEChiplet运行了操作系统，并在模拟的操作系统中运行了PARSEC 3.0基准测试程序，通过分析不同设计参数下，chiplet架构的多种评价指标，证明SEEChiplet可以成为众核chiplet架构设计空间探索的实用工具。

现有SEEChiplet为了能够实现周期级的精确模拟，需要使用串行模拟的方式对触发的事件进行逐个处理。这种方式虽然可以保证模拟的准确性，但是会导致模拟速度变慢。未来，SEEChiplet可以借鉴如parti-gem5[42]的并行模拟方案，并行模拟多个chiplet架构， 提升模拟速度。

参 考 文 献

[1]　MOORE G E. Cramming more components onto integrated

circuits[J]. *Electronics*, 1965, 38(8): 114–117.

[2] DENNARD R H, GAENSSLE F H, YU H N, *et al*. Design of ion-implanted MOSFET's with very small physical dimensions[J]. *IEEE Journal of Solid-State Circuits*, 1974, 9(5): 256–268. doi: 10.1109/JSSC.1974.1050511.

[3] HAN Yinhe, XU Haobo, LU Meixuan, *et al*. The big chip: Challenge, model and architecture[J]. *Fundamental Research*, 2023, S2667325823003709. doi: 10.1016/j.fmre.2023.10.020.

[4] CAI Jingwei, WU Zuotong, PENG Sen, *et al*. Gemini: Mapping and architecture co-exploration for large-scale DNN Chiplet accelerators[C]. 2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Edinburgh, United Kingdom, 2024: 156–171. doi: 10.1109/HPCA57654.2024.00022.

[5] 陈云霁, 蔡一茂, 汪玉, 等. 集成电路未来发展与关键问题—第347期"双清论坛(青年)"学术综述[J]. 中国科学: 信息科学, 2024, 54(1): 1–15. doi: 10.1360/SSI-2023-0356.
CHEN Yunji, CAI Yimao, WANG Yu, *et al*. Integrated circuit technology: Future development and key issues–review of the 347th "Shuangqing Forum (Youth)"[J]. *Scientia Sinica Informationis*, 2024, 54(1): 1–15. doi: 10.1360/SSI-2023-0356.

[6] 项少林, 郭茂, 蒲菠, 等. Chiplet技术发展现状[J]. 科技导报, 2023, 41(19): 113–131. doi: 10.3981/j.issn.1000-7857.2023.19.013.
XIANG Shaolin, GUO Mao, PU Bo, *et al*. Overview of the development status of Chiplet technology[J]. *Science & Technology Review*, 2023, 41(19): 113–131. doi: 10.3981/j.issn.1000-7857.2023.19.013.

[7] 厉佳瑶, 张琨, 潘权. Chiplet技术: 拓展芯片设计的新边界[J]. 集成电路与嵌入式系统, 2024, 24(2): 1–9.
LI Jiayao, ZHANG Kun, and PAN Quan. Chiplet: Expanding the innovative boundaries of chip design[J]. *Integrated Circuits and Embedded Systems*, 2024, 24(2): 1–9.

[8] MA Xiaohan, WANG Ying, WANG Yujie, *et al*. Survey on Chiplets: Interface, interconnect and integration methodology[J]. *CCF Transactions on High Performance Computing*, 2022, 4(1): 43–52. doi: 10.1007/s42514-022-00093-0.

[9] SUGGS D, SUBRAMONY M, and BOUVIER D. The AMD "Zen 2" processor[J]. *IEEE Micro*, 2020, 40(2): 45–52. doi: 10.1109/MM.2020.2974217.

[10] NAFFZIGER S, LEPAK K, PARASCHOU M, *et al*. 2.2 AMD Chiplet architecture for high-performance server and desktop products[C]. 2020 IEEE International Solid-State Circuits Conference - (ISSCC), San Francisco, USA, 2020: 44–45. doi: 10.1109/ISSCC19947.2020.9063103.

[11] EVERS M, BARNES L, and CLARK M. The AMD next-generation "Zen 3" Core[J]. *IEEE Micro*, 2022, 42(3): 7–12. doi: 10.1109/MM.2022.3152788.

[12] MUNGER B, WILCOX K, SNIDERMAN J, *et al*. Zen 4: The AMD 5nm 5.7GHz x86-64 microprocessor core[C]. 2023 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, USA, 2023: 38–39. doi: 10.1109/ISSCC42615.2023.10067540.

[13] GIANOS C. Architecting for flexibility and value with next gen Intel® Xeon® processors[C]. 2023 IEEE Hot Chips 35 Symposium (HCS), Palo Alto, USA, 2023: 1–15. doi: 10.1109/HCS59251.2023.10254694.

[14] ESPOSITO B. Intel Agilex® 9 direct RF-series FPGAs with integrated 64 Gsps data converters[C]. 2023 IEEE Hot Chips 35 Symposium (HCS), Palo Alto, USA, 2023: 1–35. doi: 10.1109/HCS59251.2023.10254707.

[15] VENTANA MICRO. Veyron V1 data center-class RISC-V processor[C]. 2023 IEEE Hot Chips 35 Symposium (HCS), Palo Alto, USA, 2023: 1–16. doi: 10.1109/HCS59251.2023.10254710.

[16] CHIRKOV G and WENTZLAFF D. Seizing the bandwidth scaling of on-package interconnect in a post-Moore's law world[C]. Proceedings of the 37th International Conference on Supercomputing, Orlando, USA, 2023: 410–422. doi: 10.1145/3577193.3593702.

[17] YANG Chongyi, ZHANG Zhendong, WANG Xiaohang, *et al*. Adaptive caching policies for Chiplet systems based on reinforcement learning[C]. 2023 IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, USA, 2023: 1–5. doi: 10.1109/ISCAS46773.2023.10181966.

[18] GADE S H, SINHA M, KUMAR M, *et al*. Scalable hybrid cache coherence using emerging links for Chiplet architectures[C]. 2022 35th International Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID), Bangalore, India, 2022: 92–97. doi: 10.1109/VLSID2022.2022.00029.

[19] MEDINA R, KEIN J, ANSALONI G, *et al*. System-level exploration of in-package wireless communication for multi-Chiplet platforms[C]. Proceedings of the 28th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 2023: 561–566. doi: 10.1145/3566097.3567952.

[20] ZHU Mingcan, SHAHAB A, KATSARAKIS A, *et al*. Invalidate or update? Revisiting coherence for tomorrow's cache hierarchies[C]. 2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Atlanta, USA, 2021: 226–241. doi: 10.1109/PACT52795.2021.00024.

[21] SHAHAB A, ZHU Mingcan, MARGARITOV A, *et al*.

Farewell my shared LLC! A case for private die-stacked DRAM caches for servers[C]. 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, Japan, 2018: 559–572. doi: 10.1109/MICRO.2018.00052.

[22] BHARADWAJ S, YIN Jieming, BECKMANN B, *et al*. Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling[C]. 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, USA, 2020: 1–6. doi: 10.1109/DAC18072.2020.9218539.

[23] IFF P, BESTA M, CAVALCANTE M, *et al*. HexaMesh: Scaling to hundreds of Chiplets with an optimized Chiplet arrangement[C]. 2023 60th ACM/IEEE Design Automation Conference (DAC), San Francisco, USA, 2023: 1–6. doi: 10.1109/DAC56929.2023.10248006.

[24] FU Yaosheng and WENTZLAFF D. PriME: A parallel and distributed simulator for thousand-core chips[C]. 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Monterey, USA, 2014: 116–125. doi: 10.1109/ISPASS.2014.6844467.

[25] LOWE-POWER J, AHMAD A M, AKRAM A, *et al*. The gem5 simulator: Version 20.0+[EB/OL]. https://arxiv.org/abs/2007.03152, 2020.

[26] UBAL R, JANG B, MISTRY P, *et al*. Multi2Sim: A simulation framework for CPU-GPU computing[C]. Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques. Minneapolis, USA, 2012: 335–344. doi: 10.1145/2370816.2370865.

[27] QURESHI Y M, SIMON W A, ZAPATER M, *et al*. gem5-X: A many-core heterogeneous simulation platform for architectural exploration and optimization[J]. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2021, 18(4): 44. doi: 10.1145/3461662.

[28] HARDAVELLAS N, SOMOGYI S, WENISCH T F, *et al*. SimFlex: A fast, accurate, flexible full-system simulation framework for performance evaluation of server architecture[J]. *ACM SIGMETRICS Performance Evaluation Review*, 2004, 31(4): 31–34. doi: 10.1145/1054907.1054914.

[29] JIANG Nan, BECKER U D, MICHELOGIANNAKIS G, *et al*. A detailed and flexible cycle-accurate network-on-chip simulator[C]. 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, USA, 2013: 86–96. doi: 10.1109/ISPASS.2013.6557149.

[30] BRKIĆ I R and JEFFREY M C M. Disintegrating manycores: Which applications lose and why?[C]. Proceedings of the 16th International Workshop on Network on Chip Architectures, Toronto, Canada, 2023: 3–8. doi: 10.1145/3610396.3618090.

[31] JEFFREY M C, SUBRAMANIAN S, YAN Cong, *et al*. A scalable architecture for ordered parallelism[C]. 2015 48th International Symposium on Microarchitecture (MICRO), Waikiki, USA, 2015: 228–241. doi: 10.1145/2830772.2830777.

[32] ZHI Haocong, XU Xianuo, HAN Weijian, *et al*. A methodology for simulating multi-Chiplet systems using open-source simulators[C]. Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication, New York, NY, USA, 2021: 18. doi: 10.1145/3477206.3477459.

[33] ORENES-VERA M, TURECI E, MARTONOSI M, *et al*. DCRA: A distributed Chiplet-based reconfigurable architecture for irregular applications[EB/OL]. https://arxiv.org/abs/2311.15443, 2024.

[34] ORENES-VERA M, TURECI E, MARTONOSI M, *et al*. MuchiSim: A simulation framework for design exploration of multi-chip Manycore systems[C]. Proceedings of the 2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Indianapolis, USA, 2024: 48–60. doi: 10.1109/ISPASS61541.2024.00015.

[35] LI Xingyu. High-performance FPGA-accelerated Chiplet modeling[D]. [Master dissertation], University of California, Berkeley, 2022.

[36] CHIRKOV G and WENTZLAFF D. SMAPPIC: Scalable multi-FPGA architecture prototype platform in the cloud[C]. Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Vancouver, Canada, 2023: 733–746. doi: 10.1145/3575693.3575753.

[37] ZHAN Xusheng, BAO Yungang, BIENIA C, *et al*. PARSEC3.0: A multicore benchmark suite with network stacks and SPLASH-2X[J]. *ACM SIGARCH Computer Architecture News*, 2017, 44(5): 1–16. doi: 10.1145/3053277.3053279.

[38] HARDAVELLAS N, FERDMAN M, FALSAFI B, *et al*. Reactive NUCA: Near-optimal block placement and replication in distributed caches[J]. *ACM SIGARCH Computer Architecture News*, 2009, 37(3): 184–195. doi: 10.1145/1555815.1555779.

[39] AWASTHI M, SUDAN K, BALASUBRAMONIAN R, *et al*. Dynamic hardware-assisted software-controlled page placement to manage capacity allocation and sharing within large caches[C]. 2009 IEEE 15th International Symposium on High Performance Computer Architecture, Raleigh, USA, 2009: 250–261. doi: 10.1109/HPCA.2009.4798260.

[40] KIM C, BURGER D, and KECKLER S W, *et al*. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches[C]. Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, USA, 2002: 211–222. doi: 10.1145/605397.605420.

[41] LI Chengeng, JIANG Fan, CHEN Shixi, *et al*. Accelerating cache coherence in Manycore processor through silicon photonic Chiplet[C]. Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD'22), San Diego, USA, 2022: 43. doi: 10.1145/3508352.3549338.

[42] CUBERO-CASCANTE J, ZURSTRAßEN N, NÖLLER J, *et al*. Parti-gem5: Gem5's timing mode parallelised[C]. 23rd International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, Samos, Greece, 2023: 177–192. doi: 10.1007/978-3-031-46077-7_12.

张聪武：男，博士生，研究方向为体系结构模拟器和异构计算.

刘 澳：男，硕士生，研究方向为体系结构模拟器和PCIe.

张 科：男，博士，正高级工程师，研究方向为计算机体系结构、异构加速、FPGA云.

常铁松：男，博士，高级工程师，研究方向为计算机体系结构和异构计算.

包云岗：男，博士，研究员，研究方向为数据中心体系结构、处理器芯片敏捷设计方法论、开源处理器芯片生态.

责任编辑：余 蓉