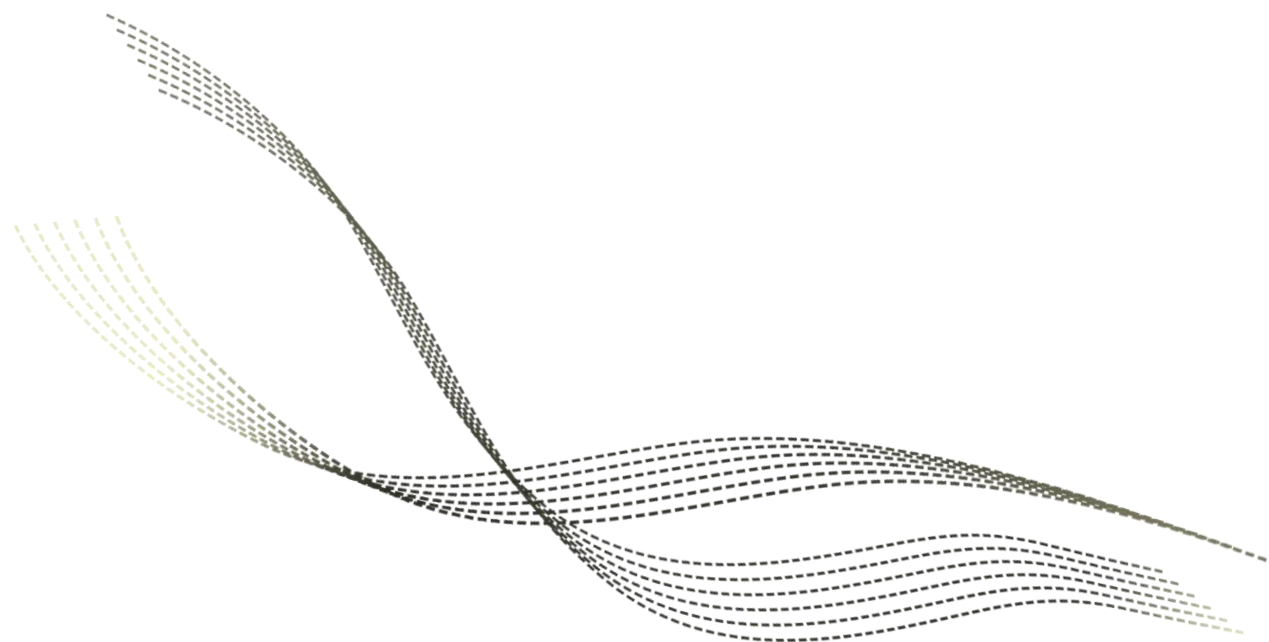


高级计算机体系结构

Advanced Computer Architecture

片上处理器与多核系统

沈明华



目录

CONTENTS

01

线程级并行

02

多核介绍

03

设计空间探索

04

从多核到众核

PART 01

线程级并行

■ 背景

□ 线程：处理器调度的基本单元

- 包含代码执行所需的上下文信息
- 同一进程中的所有线程共享代码以及数据

□ 线程可能指代

- 并行程序的一个子部分(一个子线程)
- 或一个独立程序(重量级进程)

□ 线程级并行(TLP, thread level parallel)

- 使用多个并行的执行线程协作完成一个计算任务

■ 多线程需求

- 长内存延迟无法被指令级并行完全隐藏，导致FU利用率迅速下降
 - 内存延迟：处理器等待内存传递数据的时间
- 现代处理器通常采用硬件多线程技术应对内存延迟导致的资源闲置
- 多线程处理器能并行执行多个线程的指令流，依靠：
 - 资源复制，增加多套程序计数器、寄存器...
 - 确保线程间执行上下文互不干扰
 - 增加线程切换逻辑
 - 令流水线支持线程切换



■ 多线程实现类型

□ 细粒度(交叉)多线程

- 每个时钟周期都在切换执行线程
- 但单个线程的执行会变慢

□ 粗粒度多线程

- 仅当线程遇到长延迟事件时才触发线程切换
- 但短延迟事件无法通过线程并行隐藏

□ 同时多线程(SMT, simultaneous multithreading)

- 一个时钟周期内，处理器可以同时发射多个线程的指令
- Intel的超线程技术可以让一个物理核心同时处理两个线程的指令流

■ SMT与Eckert-Mauchly奖

□ Eckert-Mauchly奖

计算机协会(ACM)和IEEE计算机协会联合宣布，华盛顿大学保罗·G·艾伦计算机科学与工程学院的教授 Susan Eggers，因“在SMT处理器架构和多处理器共享及一致性方面的开创性贡献”，荣获 2018 年Eckert-Mauchly奖。



广泛认为她是该领域的顶尖计算机架构师之一，Eggers 是该奖项 39 年历史中首位获奖女性。她于 1965 年获得电机工程学士学位(当时该领域女性占比仅 18%)，在相关领域工作近 20 年后重返校园，于 1989 年进入加州大学伯克利分校攻读博士，1999 年毕业，47 岁时在华盛顿大学开启教职生涯。

■ SMT对资源利用率的影响

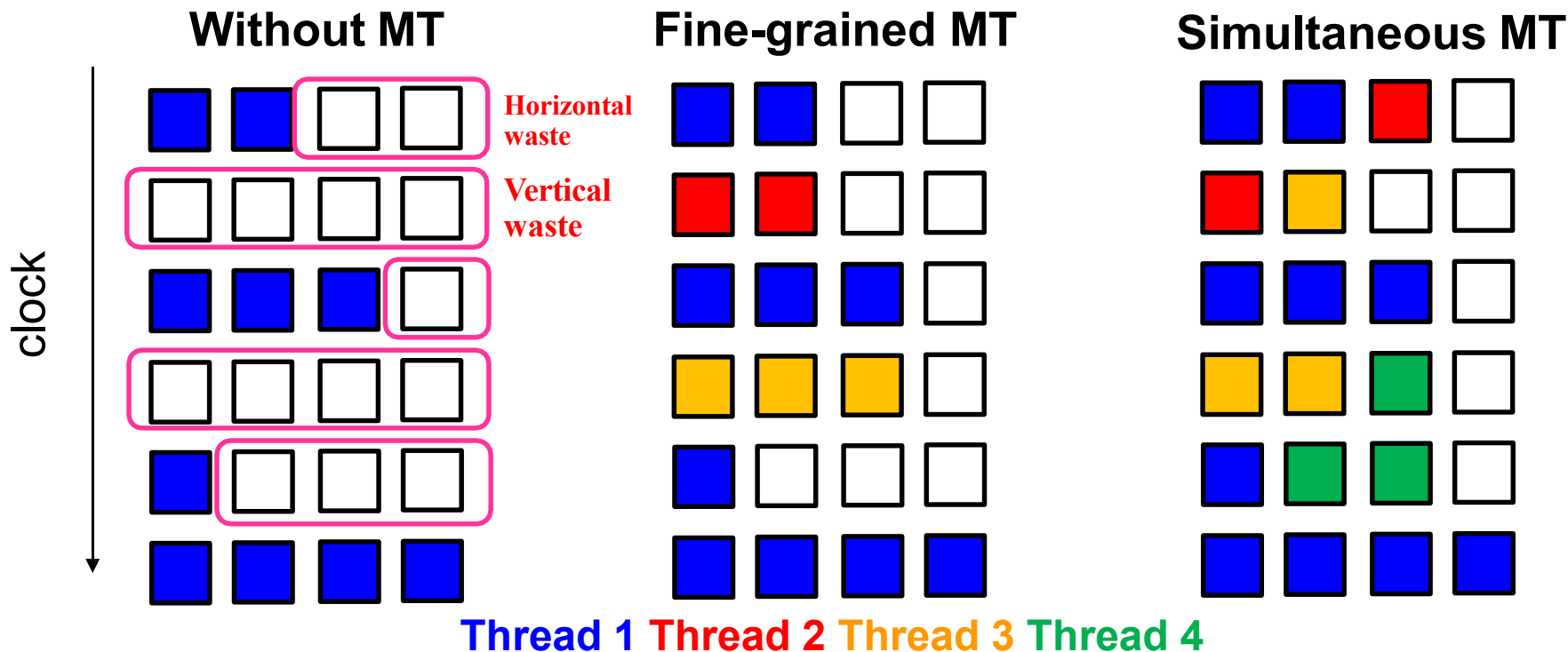
□ 多线程处理器从不同维度提升了硬件利用率

– 没有多线程会造成水平浪费和垂直浪费

➤ 水平浪费：同一周期内多个计算资源闲置

➤ 垂直浪费：同一线程在多个周期内出现资源闲置

细粒度：填补了垂直浪费，但水平浪费没解决



■ SMT对资源利用率的影响

- SMT可以充分利用独立程序之间或单个程序内部的并行性
- 指令的混合执行和调度由硬件完成(无需编译器支持) 一个CPU分成两半在跑
- SMT处理器需要复杂的硬件支持
 - 多个程序计数器
 - 每个线程需要有独立的程序计数器, 记录当前执行位置
 - 多个复杂寄存器组
 - 每个线程需要有独立的寄存器集合, 用于保存线程上下文

■ 单芯片多处理器 (Chip Multi-Processor, CMP)

□ 多处理器技术多年前就存在

- 早期并非集成在单个芯片上，而是一台机器装多个CPU

□ CMP是一种特殊的多处理器

- 将多个核心集成在单个芯片内
- 多核处理器的由来

intel 14600k

□ 总的来说多核处理器大致分为

- 共享内存多处理器
- 多指令多数据(MIMD)

■ CMP vs. SMT

□ 单芯片多核(CMP)

- 每个物理核心有独立的资源
- L1缓存、页表、程序计数器、通用寄存器是独立的
- 但L2缓存可能是共享的

□ 同步多线程(SMT)

- 多线程共享处理器的资源
- 缓存和页表共享，但程序计数器和通用寄存器独立

□ 超线程(HT, hyper threading)

- 英特尔的SMT技术。一个核心运行两个线程，性能大致是单线程的1.4倍
 - 不过在主打能耗的下一代Ultra系列处理器上被移除
 - 原因是SMT复杂的控制电路增加功耗

■ 并行性对比

□ 并行粒度

– 超标量

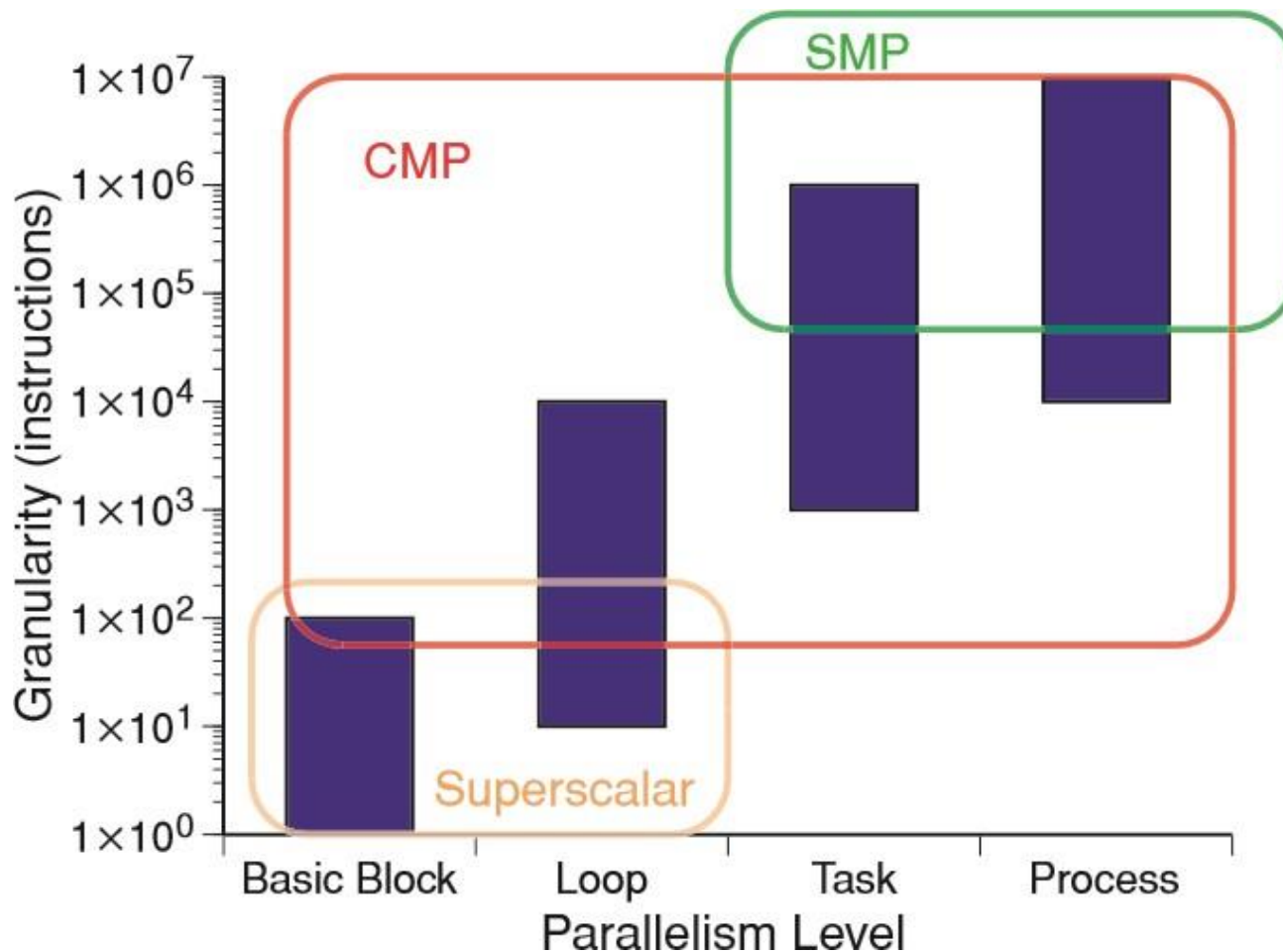
- 指令级并行
- 粒度最细

– CMP

- 并行粒度覆盖最广

– 多处理器系统 (SMP)

- 任务级和进程级并行
- 并行粒度最粗



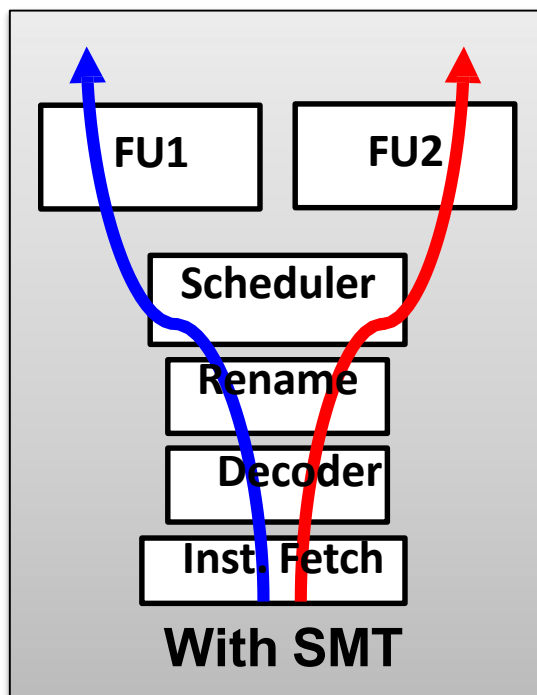
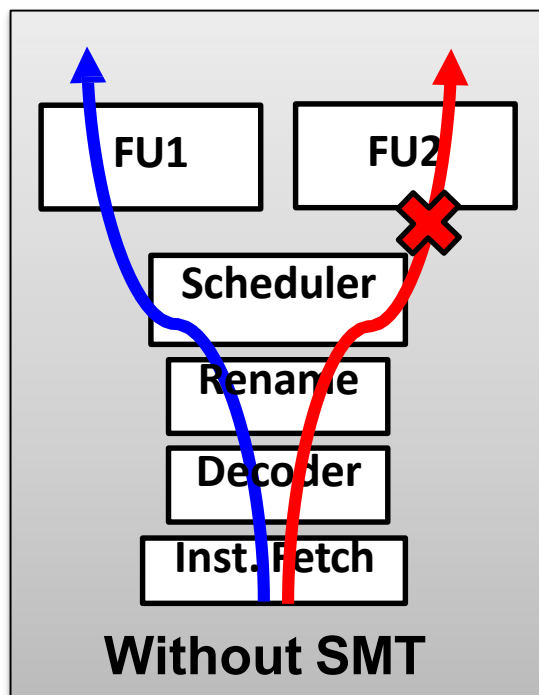
PART 02

多核介绍

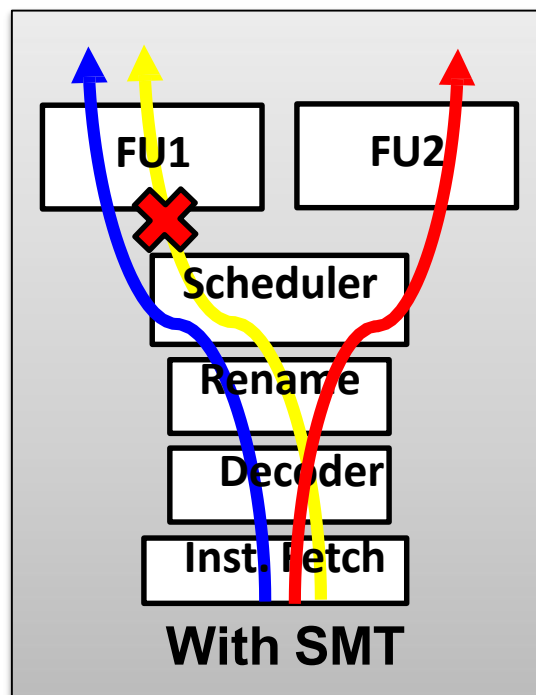
■ 为什么需要多核？

□ 单核的超标量处理器无法充分利用线程级并行

- 无SMT：同一时间仅一个线程能执行，配备的其他FU闲置
- 有SMT：硬件资源的线程独占性限制了并行度
 - SMT不能支持争用相同FU的两个线程并行



占用不同FU，可以并行

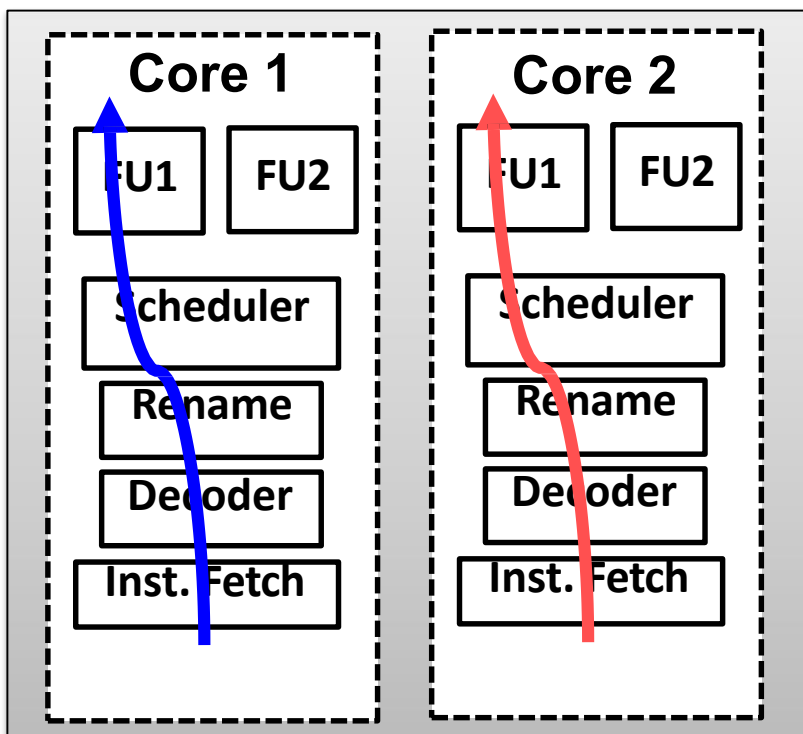


争用FU1，不可以并行

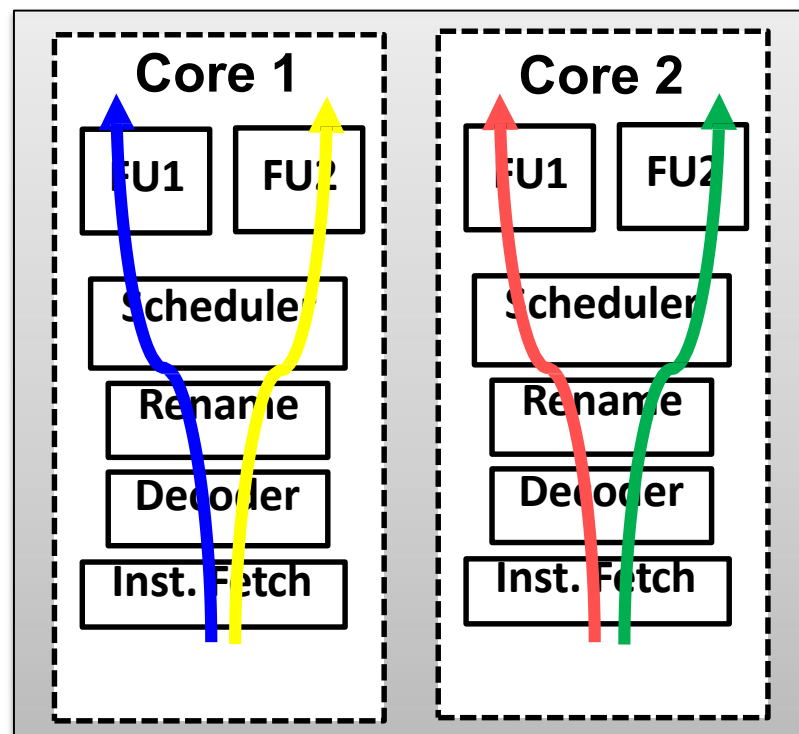
■ 为什么需要多核？

□ 多核架构能利用线程级并行

- 每个核心可独立运行线程
- SMT+多核：每个核心内部通过SMT同时调度多个线程，更充分并行



争用FU1的线程运行在不同核心上



搭配SMT允许四线程并行

■ 为什么需要多核？

□ 单核SMT

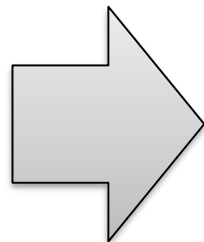
- 本质上还是依赖指令级并行(如超标量执行)
- 仅能通过提升时钟频率来加速任务执行
- 会大幅增加功耗、散热需求。设计困难，设计更加耗时，验证难度更大

□ 多核解决方案

- 天然支持线程级并行
- 单芯片集成多个核心
- 核心在最高效的频率区间运行(无需依赖高频时钟提升性能)
- 复用经过验证的处理器设计，显著降低制造成本

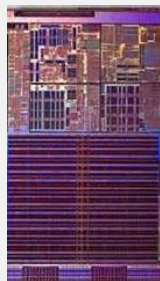
■ 多核的能耗优势

当电压下降15%时



频率降低	功耗降低	性能降低
15%	45%	10%

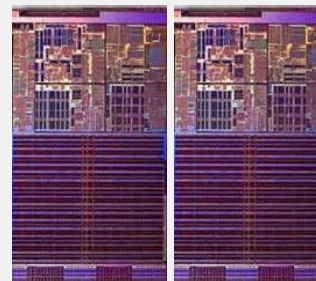
Single Core



面积 = 1
电压 = 1
频率 = 1
功耗 = 1
性能 = 1

V.S.

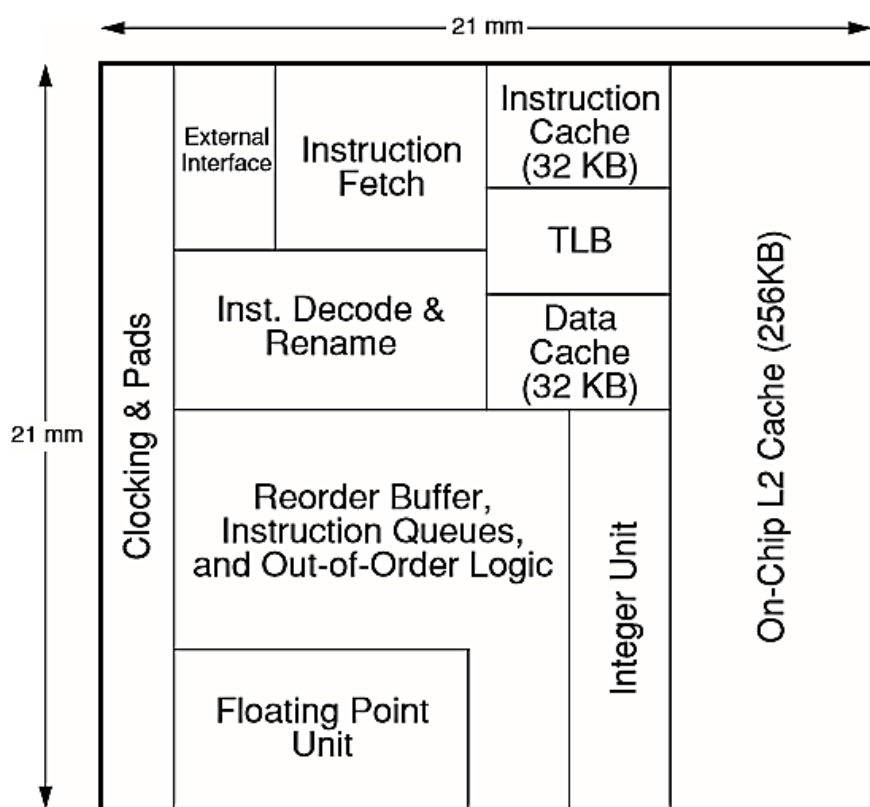
Dual Core



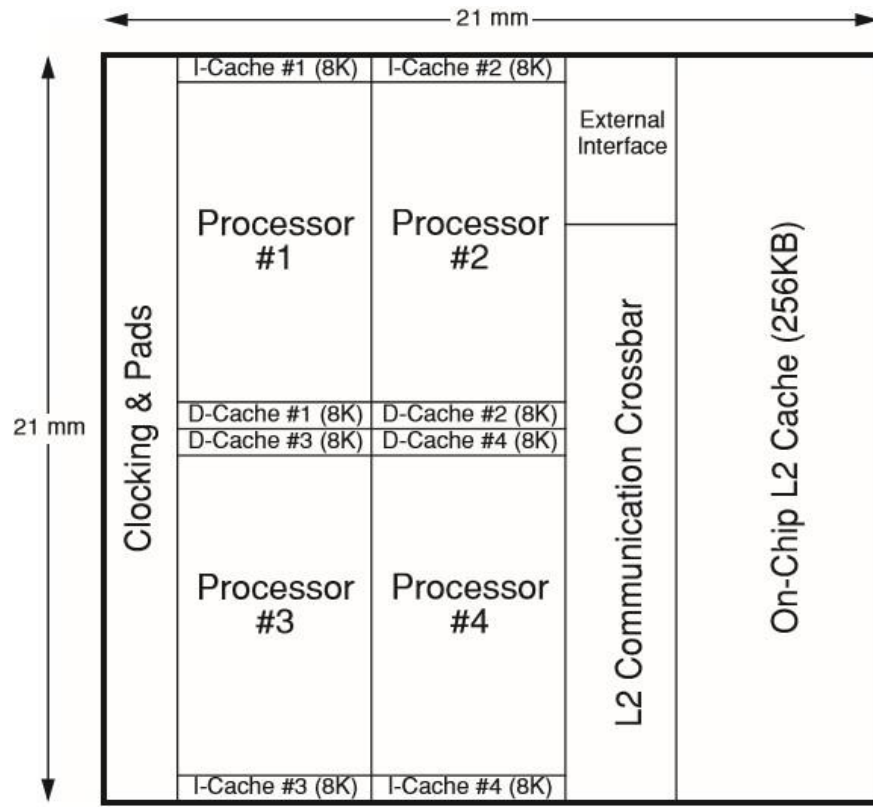
面积 = 2
电压 = 0.85
频率 = 0.85
功耗 = 1.1
性能 ~ 1.8

■ 超标量处理器 vs. 单芯片多处理器

- 假设以下两种处理器的加工工艺、芯片面积、片外资源、时钟频率都相同



6发射超标量处理器



4核单芯片多处理器

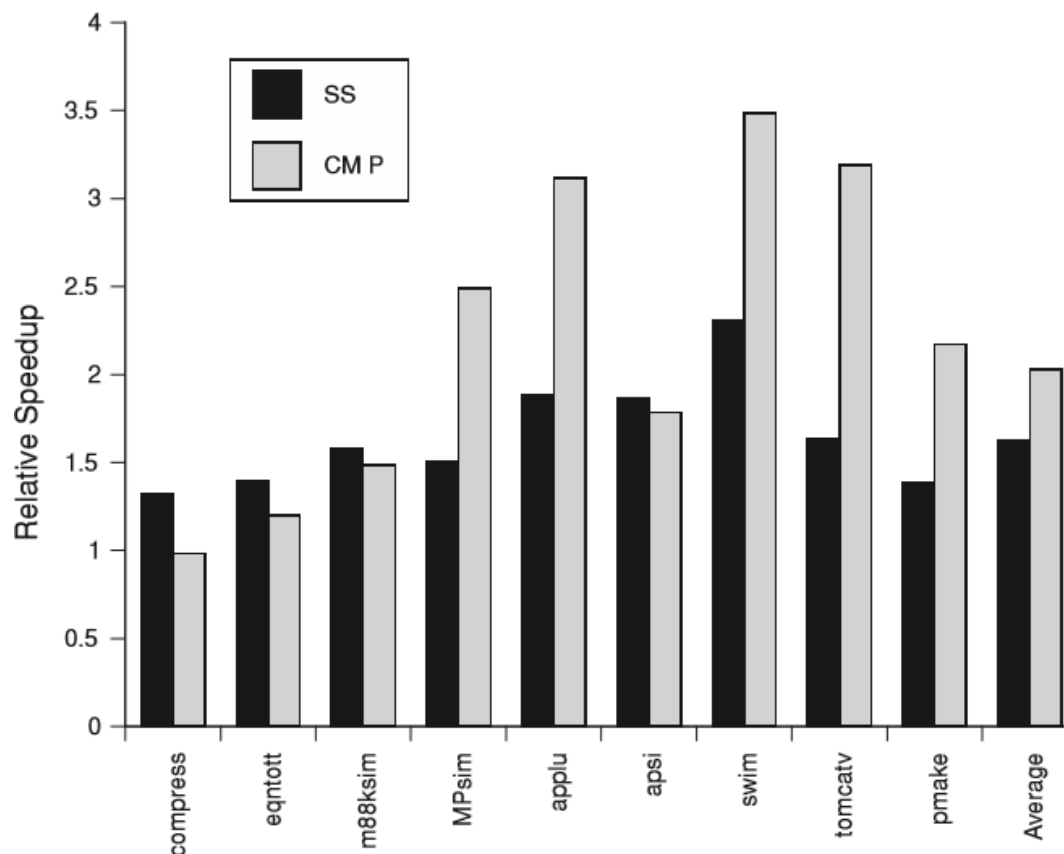
■ 超标量处理器 vs. 单芯片多处理器

□ 大型超标量处理器

- 充分发掘单线程的指令级并行度
- 程序开发难度小
- 单线程优化主要由硬件/编译器完成

□ 单芯片多处理器

- 结合指令级并行和细粒度线程级并行
- 需程序员开发多线程程序
- 适用于具备大量并行任务的程序

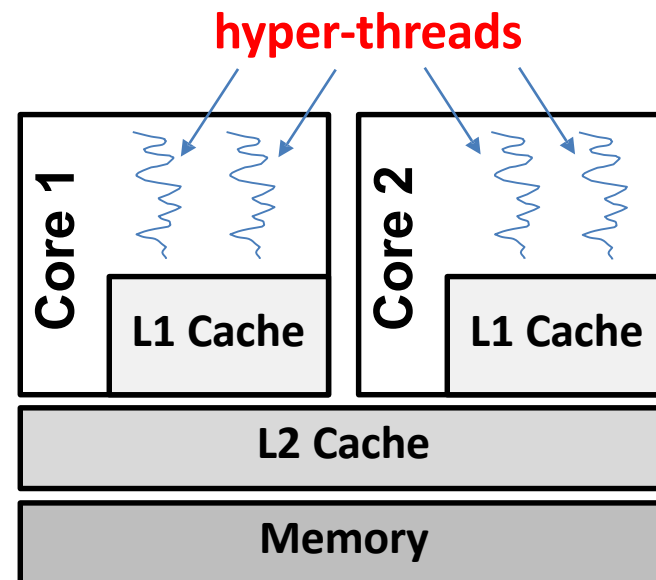


超标量处理器SS与多核处理器CMP性能对比(基线为2发射超标量处理器)

■ 经典多核缓存组织架构

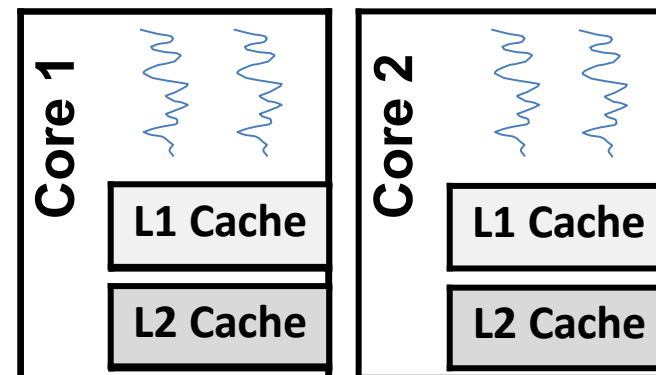
□ 私有L1 + 共享L2

- 代表作：双核英特尔至强处理器



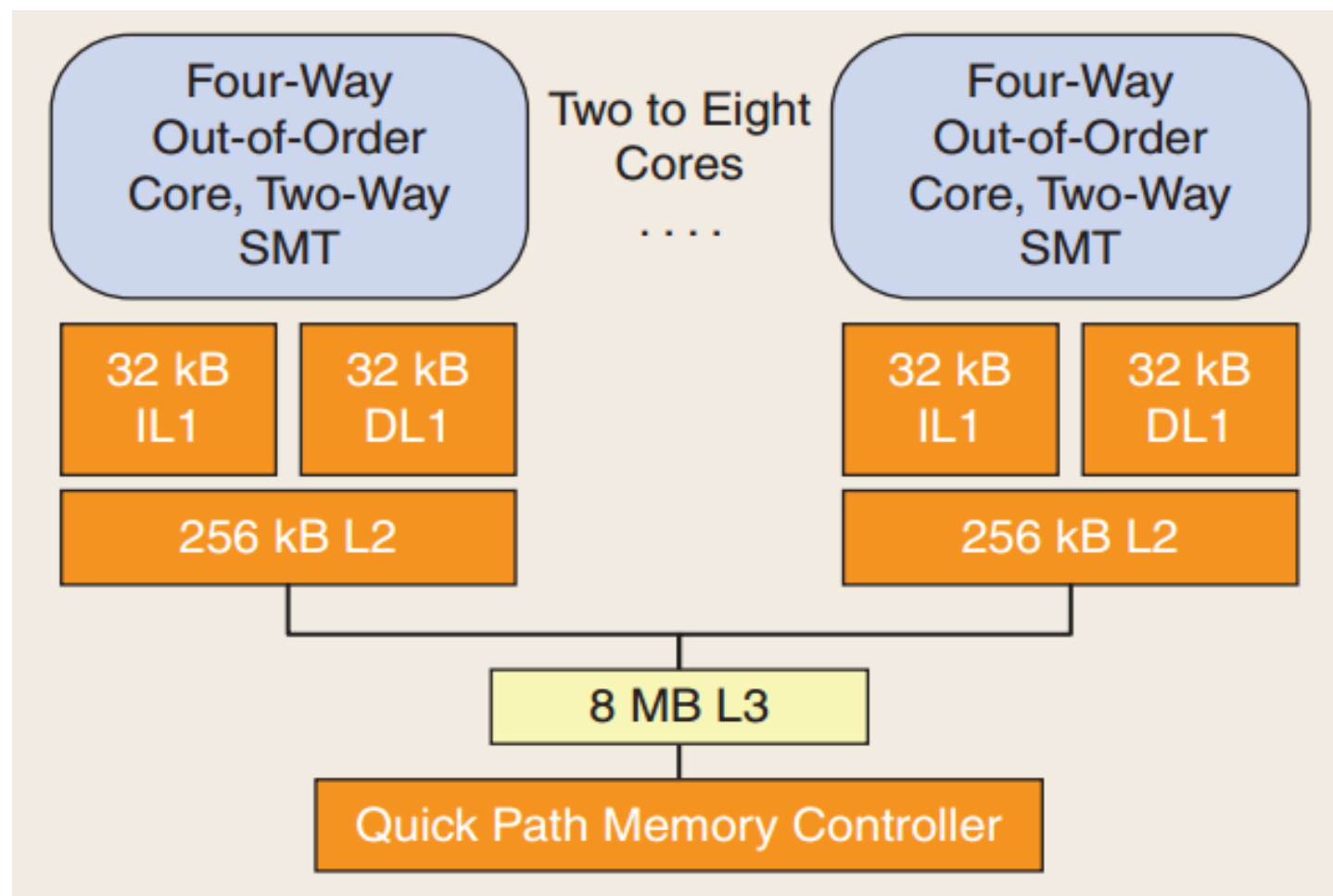
□ 私有L1 + 私有L2

- 代表作：AMD速龙处理器
- 代表作：英特尔奔腾D处理器



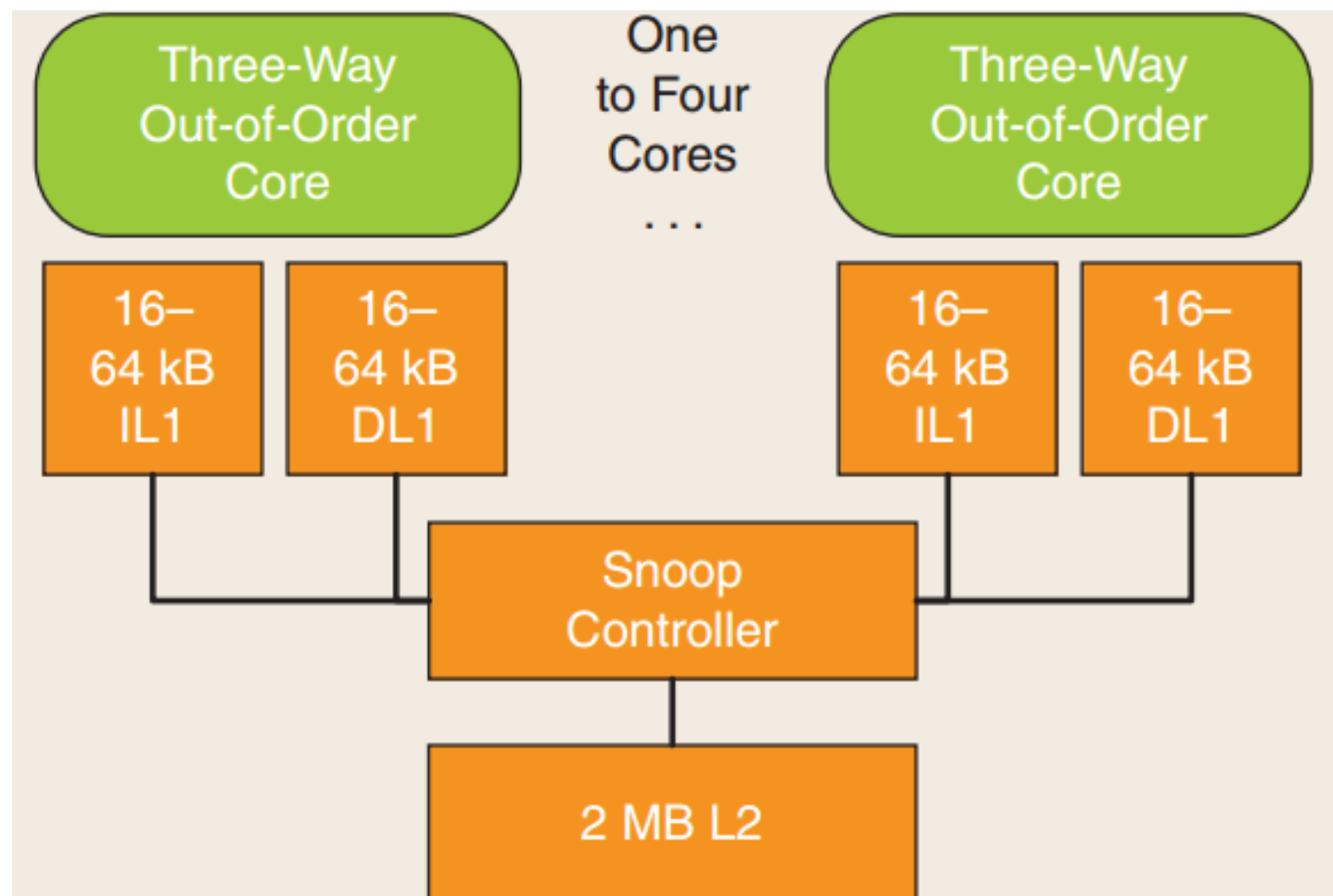
■ 多核示例

□ Intel i7



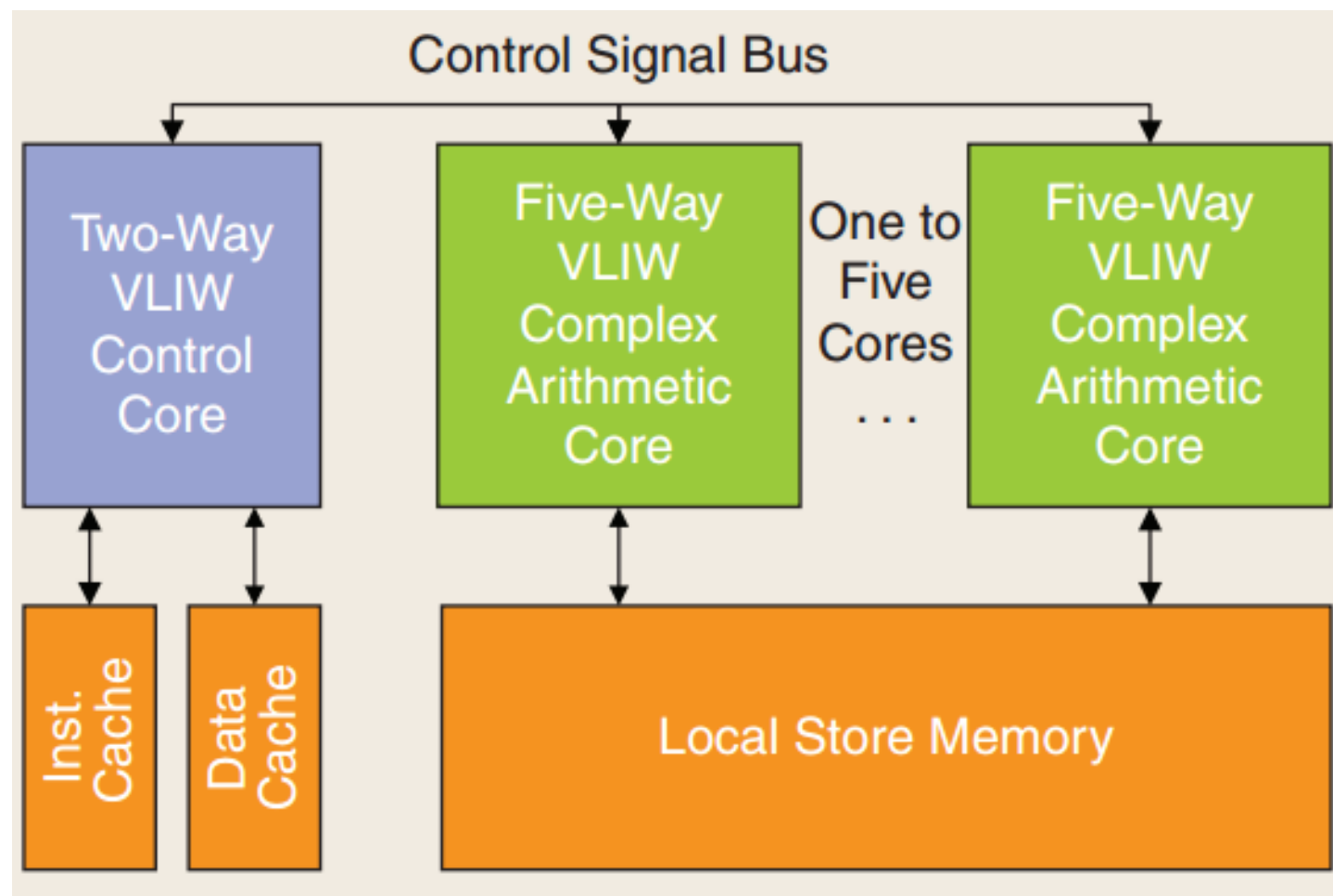
■ 多核示例

□ ARM Cortex-A9



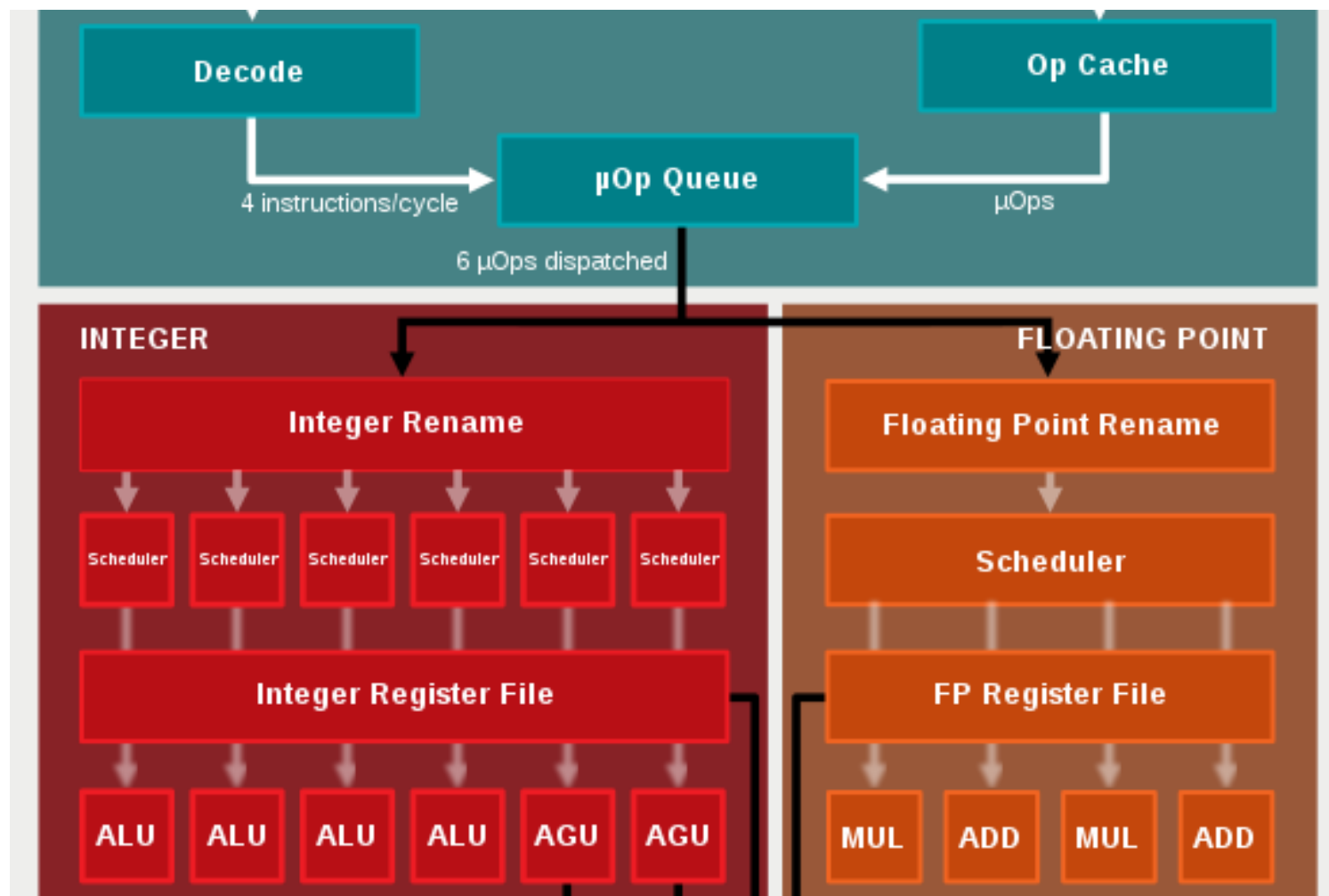
■ 多核示例

□ Silicon Hive HiveFlex CSP2x00



■ 多核示例

□ AMD's High-Performance Zen CPU



■ 通用服务器/移动/嵌入式设备多核处理器架构

- 下述核心都使用了传统的微架构，且都以某个强大的单处理器为蓝本
 - 多核设计复用单核经典设计

	ISA	Micro arch	# of Core	Coherence	Interconnect
AMD Phenom	X86	3-way OOO	4	Directory	P2P
Intel Core i7	X86	4-way OOO	2~8	Broadcast	P2P
Sun Niagara	SPARC	2-Way In-order	8	Directory	Crossbar
Intel Atom	X86	2-way In-order	1~2	Broadcast	Bus
ARM Cortex A9	ARM	3-way OOO	1~4	Broadcast	Bus
XMOS XS1-G4	XCORE	1-way in-order	4	None	Crossbar

■ 高性能多核/众核处理器

□ 下述核心使用专业化高性能设计

- 侧重多核并行，核心数上百；部分放弃缓存一致性以换取更高的并行度
- 微架构和互联方式高度定制化，是任务特化型处理器

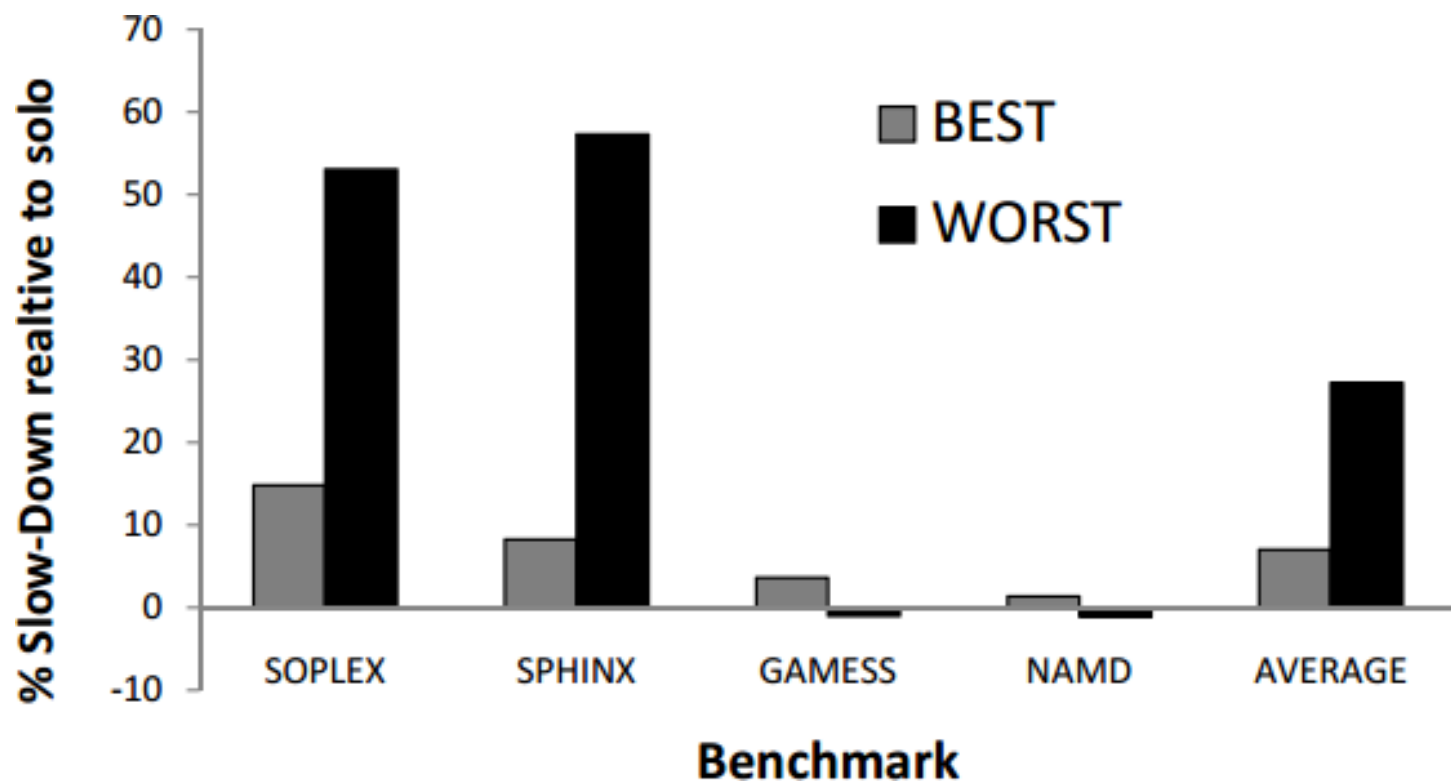
	ISA	Micro arch	# of Core	Coherence	Interconnect
AMD Radeon	N/A	5-way VLIW	160	None	N/A
NVIDIA G200	N/A	1-way In-order	240	None	N/A
Intel Larrabee	X86	2-Way In-order	Up to 48	Broadcast	Bidirectional Ring
IBM Cell	POWER	2-way In-order	8 SPU's	None	Bidirectional Ring
Microsoft Xenon	POWER	2-way In-order	3	Broadcast	Crossbar

■ 多核架构存在的挑战

- 共享资源管理
- 权衡指令级并行与线程级并行
- 匹配任务粒度与核心数
- 片上/片外带宽需求
- 延迟(执行、内存、缓存)降低
- 电源管理的多域问题
- 线程/核心间资源划分, 缓存一致性问题
- 片上互联问题

■ 多核资源竞争

- CMP上的核心并非独立的处理器，而是片上系统的一部分，与其他核心共享资源
 - 因此线程会竞争最后一级缓存(LLC)、MC、总线、预取硬件等关键资源
 - 资源调度策略和应用特性是影响多核性能的关键因素



在Intel Xeon X3565 四核处理器(两个核心共享一个LLC)上，针对SPEC CPU2006应用的两种不同调度策略，与单独运行相比，性能下降程度

PART 03

设计空间探索

■ 设计空间：选择核心

□ 如何混合不同类型的处理器核心，来设计一个强大的处理器

- 少量复杂、重量级核心
 - 侧重于降低单线程运行时间，用效率增加处理器吞吐量。单核面积大
- 大量简单、轻量级核心
 - 侧重于降低核心面积，用数量增加处理器吞吐量。单线程运行时间较长

□ 通用处理器核心

- 面积小、能效高核心 (面向移动设备、嵌入式场景)
- 面积大、高性能核心 (面向高性能计算场景)

□ 专用核心

- 针对特定类任务的加速器

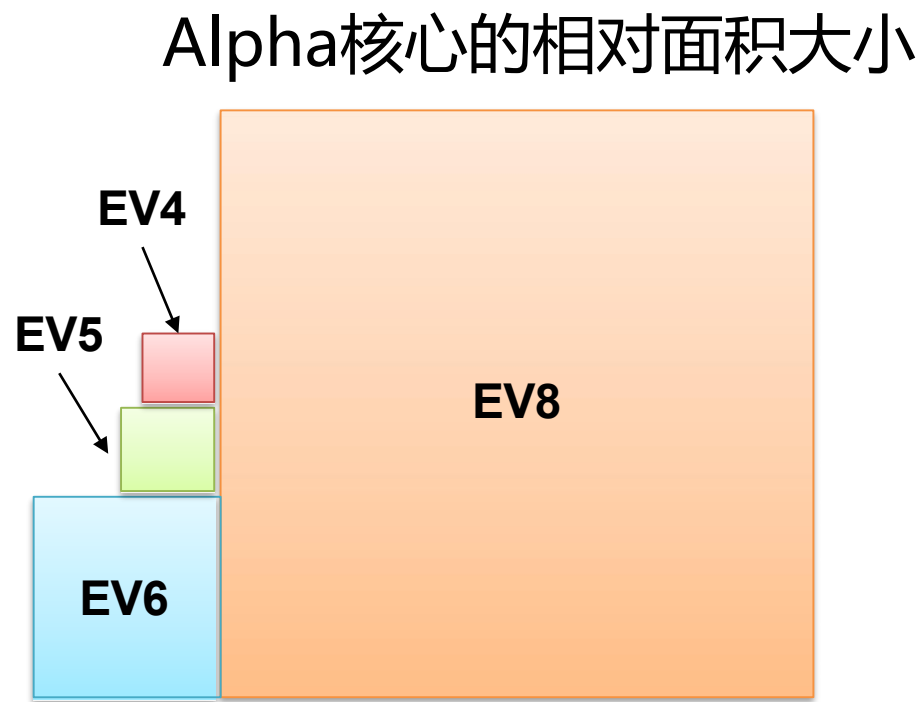
■ 设计空间：选择核心

□ 例如：Alpha处理器的EV系列特性

- 组合以下不同核心得到一个峰值性能强，面积小，低功耗能效高的处理器
 - EV4：早期CMOS微处理器
 - EV5：在片上集成二级缓存
 - EV6：支持乱序执行
 - EV8：引入SMT

Alpha处理器核心的功耗以及相对性能

Core	Peak Power	Avg. Power	Norm. Perf
EV4	5.0	3.7	1.00
EV5	9.8	6.9	1.30
EV6	17.8	10.7	1.87
EV8	92.88	46.4	2.14



■ 单指令集异构多处理器

- 以上又被称为非对称多处理器(asymmetric chip multiprocessors)
 - 每个核心能力不同
 - 可以把每个应用与最适合其性能需求的核心相匹配
 - 更高效地利用处理器资源，满足不同工作负载需求
- 非对称多处理器可以进一步划分为：单指令集异构与多指令集异构
 - 单指令集CMP由一组异构处理器核心组成，但所有核心都可执行相同的指令集
 - 通过改变以下特性实现性能和功耗的不同偏向
 - 超标量宽度、基准频率
 - 缓存大小
 - 其它特性 (顺序执行、乱序执行...)

■ 多指令集异构多处理器

- 多指令集，相比单指令集更灵活，探索空间更大
 - ARM Thumb、Intel x86-64、DEC Alpha
- 充分利用多指令集异构优势面临多重挑战
 - 应用程序需要可以在核心之间自由迁移，但异构指令集的核心间迁移困难重重
 - 挑战：需要保存带特定指令集信息的运行时状态
 - 迁移进程涉及代价高昂的程序状态切换，通常包含以下几步
 - 调度：选定一个核心将进程迁移上去，需综合考虑迁移成本、迁移收益
 - 页表修改：修改页表的映射关系
 - 二进制翻译：把进程的运行时状态翻译成可迁移的特殊格式
 - 状态恢复：恢复进程的运行时状态，从而在新核心上运行进程

■ 异构多核的分类

Same ISA	Different ISA	
	Same Cores	Different Cores
	同构多核/众核 代表作：普通处理器	多指令集 异构处理器 代表作：intel的Ultra
	??	单指令集 异构处理器 代表作：AMD Zen5

分为性能核P核、能效核E核。其中E核不支持AVX512指令，而P核支持。因此必须搭配windows11的大小核调度

分为大核Zen5、小核Zen5c。Zen5大核共享16MB的L2缓存，最高频率5.3G。Zen5c小核共享8MB的L2缓存，最高频率3.3G。

PART 04

从多核到众核

■ 多核 vs. 众核

□ 众核 (Manycore)

- 大量核心
- 为高并行性特化
- 更高吞吐量
- 单线程性能较低

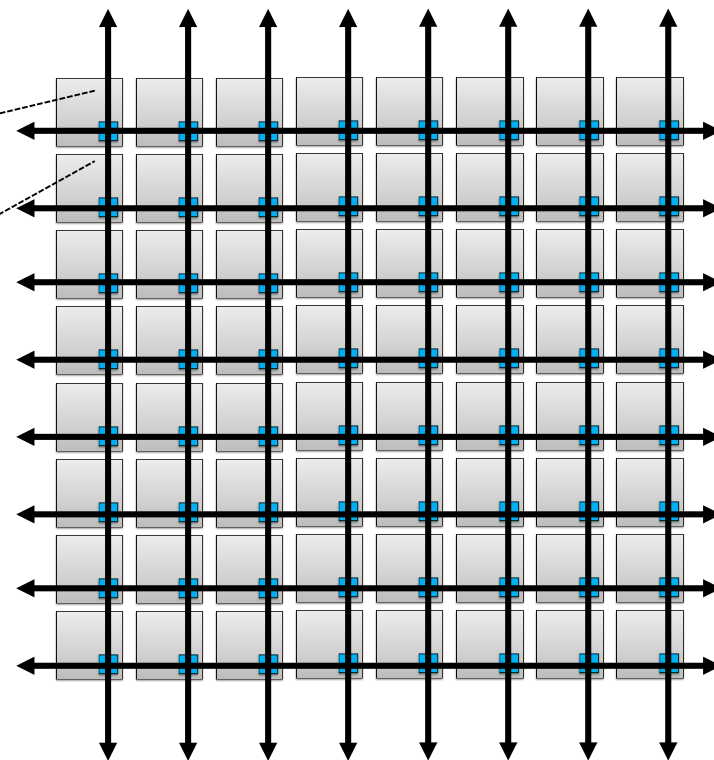
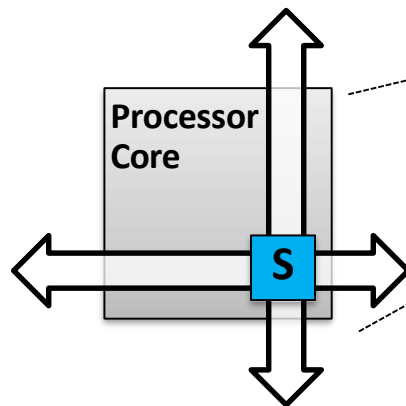
□ 多核 (Multicore)

- 少量核心
- 并行和串行代码都进行优化
- 采用乱序执行(OOO)、更深的流水线等技术
- 更强调单线程性能

■ Tiler的Tile众核架构

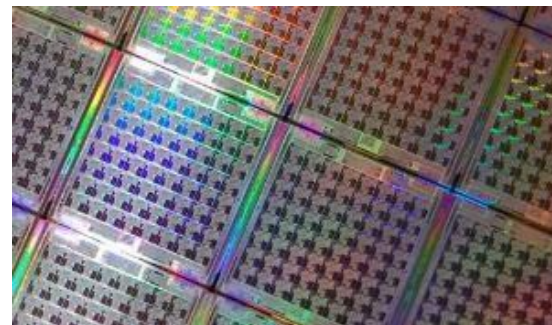
□ 支持大规模扩展

- 重复排列Tile
- 进入十亿晶体管架构时代



□ Tile = core + switch

- 每个Tile兼具计算和通讯功能
- 模块化、可扩展、保持缓存一致性
- 设计周期短
 - 复用tile的模块化设计



■ 英特尔集成众核架构 (MIC)

□ Intel Xeon Phi 协处理器

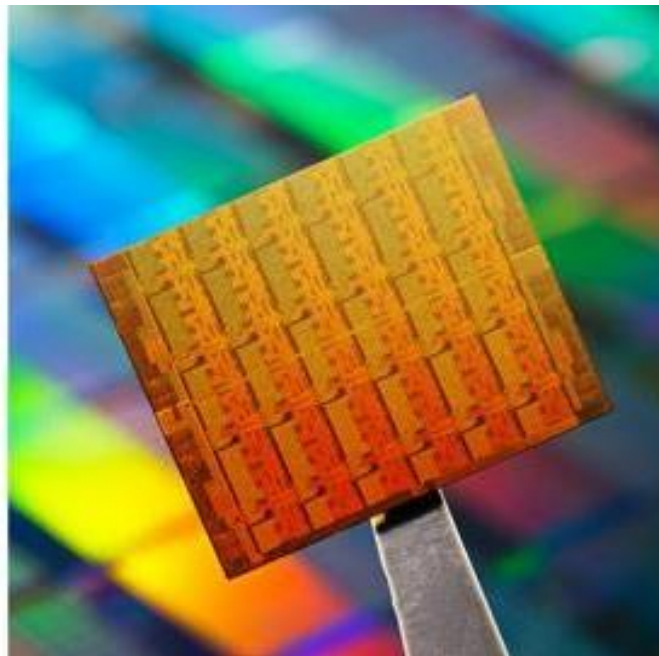
- 至多集成61个顺序执行核心的众核处理器
- 每个核心均支持4个超线程
- 核心间的L2缓存具有缓存一致性
- 自带操作系统，占用一个核心
- 提供全系统中性能-功耗比最优的协处理器资源



采用PCIe卡形态

■ Intel 单芯片云计算 (SCC)

- 在芯片上集成计算机云(SCC, single-chip cloud computer)
 - 是一种众核架构
 - SCC是云数据中心的微缩模型
- SCC没有硬件支持的缓存一致性，消息传递是其主要编程范式



感谢！
