

COMPUTER ORGANIZATION AND ARCHITECTURE (COA)

EET 2211
4TH SEMESTER – CSE & CSIT
CHAPTER 18, LECTURE 37
By Ms. Arya Tripathy

MULTICORE COMPUTERS

(Already covered, just go
through)

(Already covered, just go
nth row)

TOPICS TO BE COVERED

➤ HARDWARE PERFORMANCE ISSUES

- Increase in Parallelism and Complexity
- Power Consumption

➤ SOFTWARE PERFORMANCE ISSUES

- Software on Multicore

➤ MULTICORE ORGANIZATION

- Levels of Cache
- Simultaneous Multithreading

➤ HETEROGENEOUS MULTICORE ORGANIZATION

- Different Instruction Set Architectures
- Equivalent Instruction Set Architecture

LEARNING OBJECTIVES

- ❖ Understand the **hardware performance issues that have driven the move to** multicore computers.
- ❖ Understand the **software performance issues posed by the use of multithreaded** multicore computers.
- ❖ Present an overview of the two principal approaches to **heterogeneous multicore organization.**

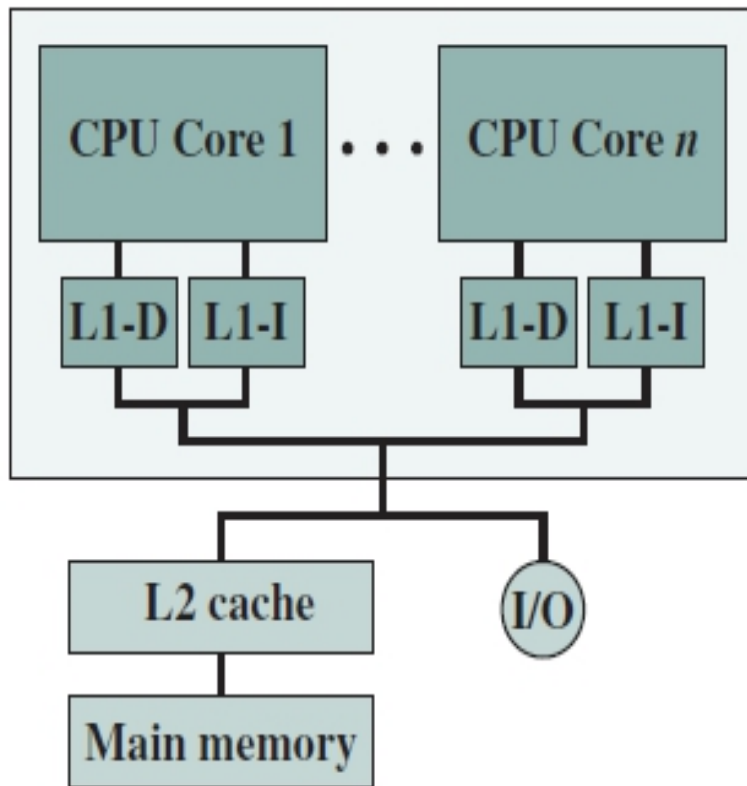
✓ MULTICORE ORGANIZATION

At a top level of description, the main variables in a multicore organization are as follows:

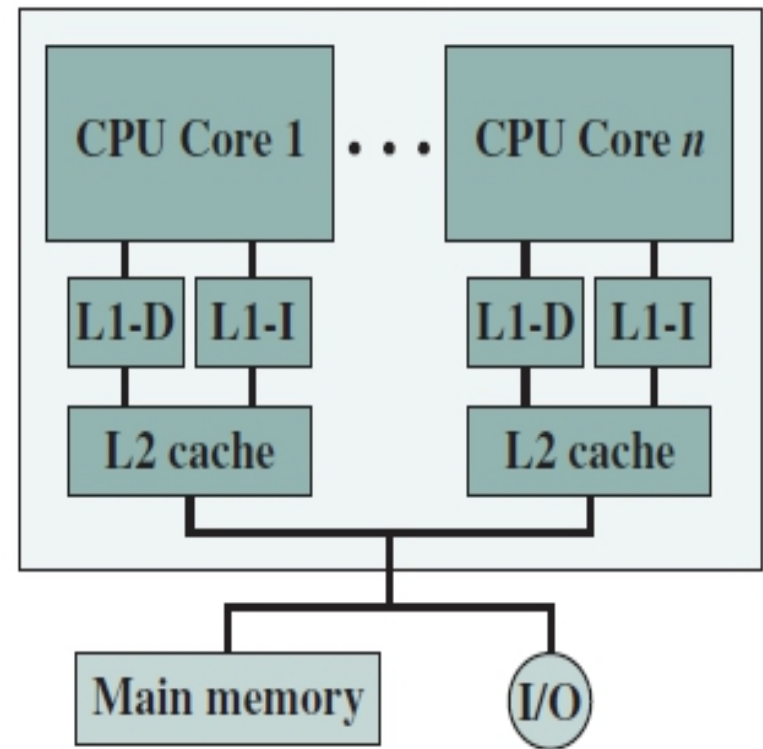
- ❑ The number of core processors on the chip
- ❑ The number of levels of cache memory
- ❑ How cache memory is shared among cores
- ❑ Whether simultaneous multithreading (SMT) is employed
- ❑ The types of cores

Levels of Cache

There are four general organizations for multicore systems.



(a) Dedicated L1 cache



(b) Dedicated L2 cache

In Figure (a) organization, the only on-chip cache is L1 cache, with each core having its own dedicated L1 cache.

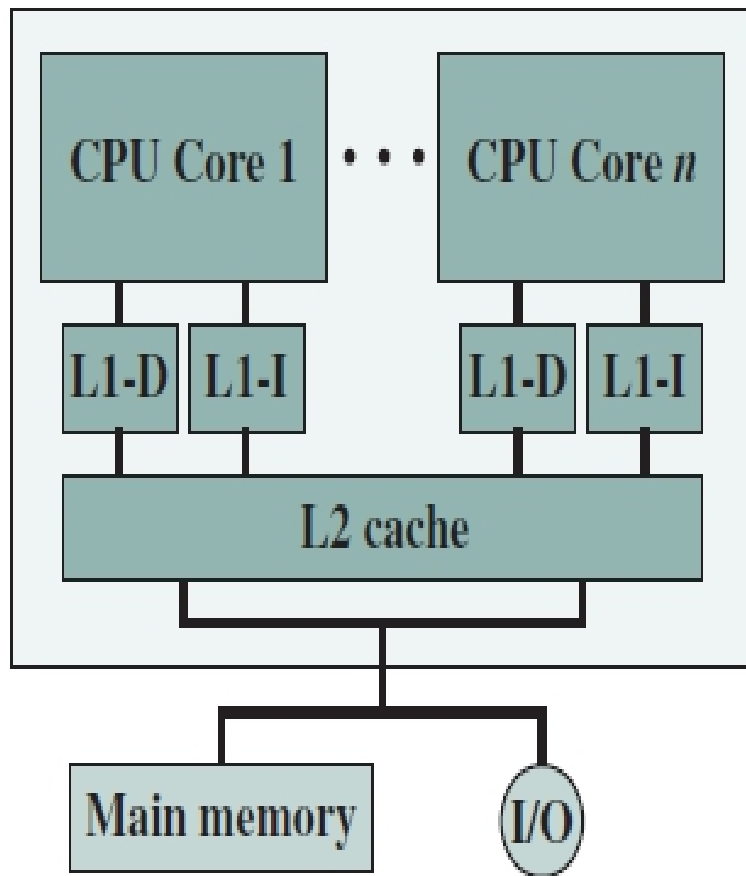
Almost invariably, the L1 cache is divided into instruction and data caches for performance reasons, while L2 and higher-level caches are unified.

An example of this organization is the ARM11 MPCore.

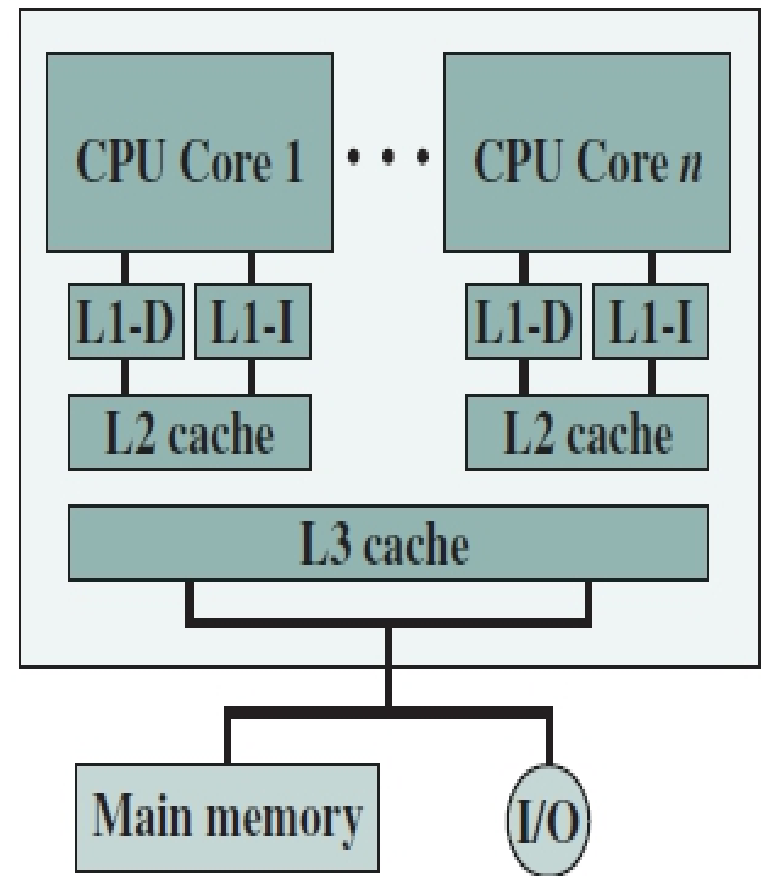
The organization of Figure (b) is also one in which there is no on-chip cache sharing.

In this, there is enough area available on the chip to allow for L2 cache.

An example of this organization is the AMD Opteron.



(c) Shared L2 cache



(d) Shared L3 cache

Figure (c) shows a similar allocation of chip space to memory, but with the use of a shared L2 cache.

The Intel Core Duo has this organization.

Finally, as the amount of cache memory available on the chip continues to grow, performance considerations dictate splitting off a separate, shared L3 cache (Figure (d)), with dedicated L1 and L2 caches for each core processor.

The Intel Core i7 is an example of this organization.

❖ The use of a shared higher-level cache on the chip has several advantages over exclusive reliance on dedicated caches:

1. Constructive interference can reduce overall miss rates.
2. A related advantage is that data shared by multiple cores is not replicated at the shared cache level.
3. With proper line replacement algorithms, the amount of shared cache allocated to each core is dynamic, so that threads that have less locality (larger working sets) can employ more cache.
4. Inter-core communication is easy to implement, via shared memory locations.
5. The use of a shared higher-level cache confines the cache coherency problem to the lower cache levels, which may provide some additional performance advantage.

❖ A potential advantage to having only dedicated L2 caches on the chip is that each core enjoys more rapid access to its private L2 cache. This is advantageous for threads that exhibit strong locality.

SIMULTANEOUS MULTITHREADING

- ❖ Another organizational design decision in a multicore system is whether the individual cores will implement **simultaneous multithreading (SMT)**.
- ❖ For example, the Intel Core Duo uses pure superscalar cores, whereas the Intel Core i7 uses SMT cores.
- ❖ SMT has the effect of scaling up the number of hardware-level threads that the multicore system supports.
- ❖ Thus, a multicore system with four cores and SMT that supports four simultaneous threads in each core appears the same to the application level as a multicore system with 16 cores.
- ❖ As software is developed to exploit parallel resources, an SMT approach appears to be more attractive than a purely superscalar approach.

✓ HETEROGENEOUS MULTICORE ORGANIZATION

❖ As clock speeds and logic densities increase, designers must balance many design elements in attempts to maximize performance and minimize power consumption.

❖ It can be done by the following approaches:

1. Increase the percentage of the chip devoted to cache memory.
2. Increase the number of levels of cache memory
3. Change the length and functional components of the instruction pipeline.
4. Employ simultaneous multithreading
5. Use multiple cores


✓❖ To achieve better results, in terms of performance and/or power consumption, an increasingly popular design choice is **heterogeneous multicore organization**, which refers to a processor chip that includes more than one kind of core.

❖ In this section, we look at two approaches to heterogeneous multicore organization.

Different Instruction Set Architectures

Typically, this involves mixing conventional cores, referred to in this context as CPUs, with specialized cores optimized for certain types of data or applications.

There are two primary examples which we need to look at.

 **CPU/GPU MULTICORE**: The most prominent trend in terms of heterogeneous multicore design is the use of both CPUs and graphics processing units (GPUs) on the same chip.

GPUs are characterized by the ability to support thousands of parallel execution threads. Thus, GPUs are well matched to applications that process large amounts of vector and matrix data.

To deal with the diversity of target applications in today's computing environment, multicore containing both GPUs and CPUs has the potential to enhance performance.

This heterogeneous mix, however, presents issues of coordination and correctness.

~~CPU~~/**DSP MULTICORE:** Another common example of a heterogeneous multicore chip is a mixture of CPUs and digital signal processors (DSPs).

A DSP provides ultra-fast instruction sequences (shift and add; multiply and add), which are commonly used in math-intensive digital signal processing applications.

DSPs are used to process analog data from sources such as sound, weather satellites, and earthquake monitors.

Signals are converted into digital data and analyzed using various algorithms such as Fast Fourier Transform.

DSP cores are widely used in myriad devices, including cellphones, sound cards, fax machines, modems, hard disks, and digital TVs.

~~Equivalent~~ Instruction Set Architectures

Another recent approach to heterogeneous multicore organization is the use of multiple cores that have equivalent ISAs but vary in performance or power efficiency.

The leading example of this is ARM's big- Little architecture.

It includes a multicore processor chip containing two high performance Cortex-A15 cores and two lower-performance, lower-power-consuming Cortex-A7 cores.

The A7 cores handle less computation intense tasks, such as background processing, playing music, sending texts, and making phone calls.

The A15 cores are invoked for high intensity tasks, such as for video, gaming, and navigation.

Across a range of benchmarks, the Cortex-A15 delivers roughly twice the performance of the Cortex-A7 per unit MHz, and the Cortex-A7 is roughly three times as energy efficient as the Cortex-A15 in completing the same workloads.

The big-Little architecture is aimed at the smart phone and tablet market.

Ch 18 Review Questions

- 18.1** Summarize the differences among simple instruction pipelining, superscalar, and simultaneous multithreading.
- 18.2** Give several reasons for the choice by designers to move to a multicore organization rather than increase parallelism within a single processor.
- 18.3** Why is there a trend toward giving an increasing fraction of chip area to cache memory?
- 18.4** List some examples of applications that benefit directly from the ability to scale throughput with the number of cores.
- 18.5** At a top level, what are the main design variables in a multicore organization?
- 18.6** List some advantages of a shared L2 cache among cores compared to separate dedicated L2 caches for each core.

