Lecture 11

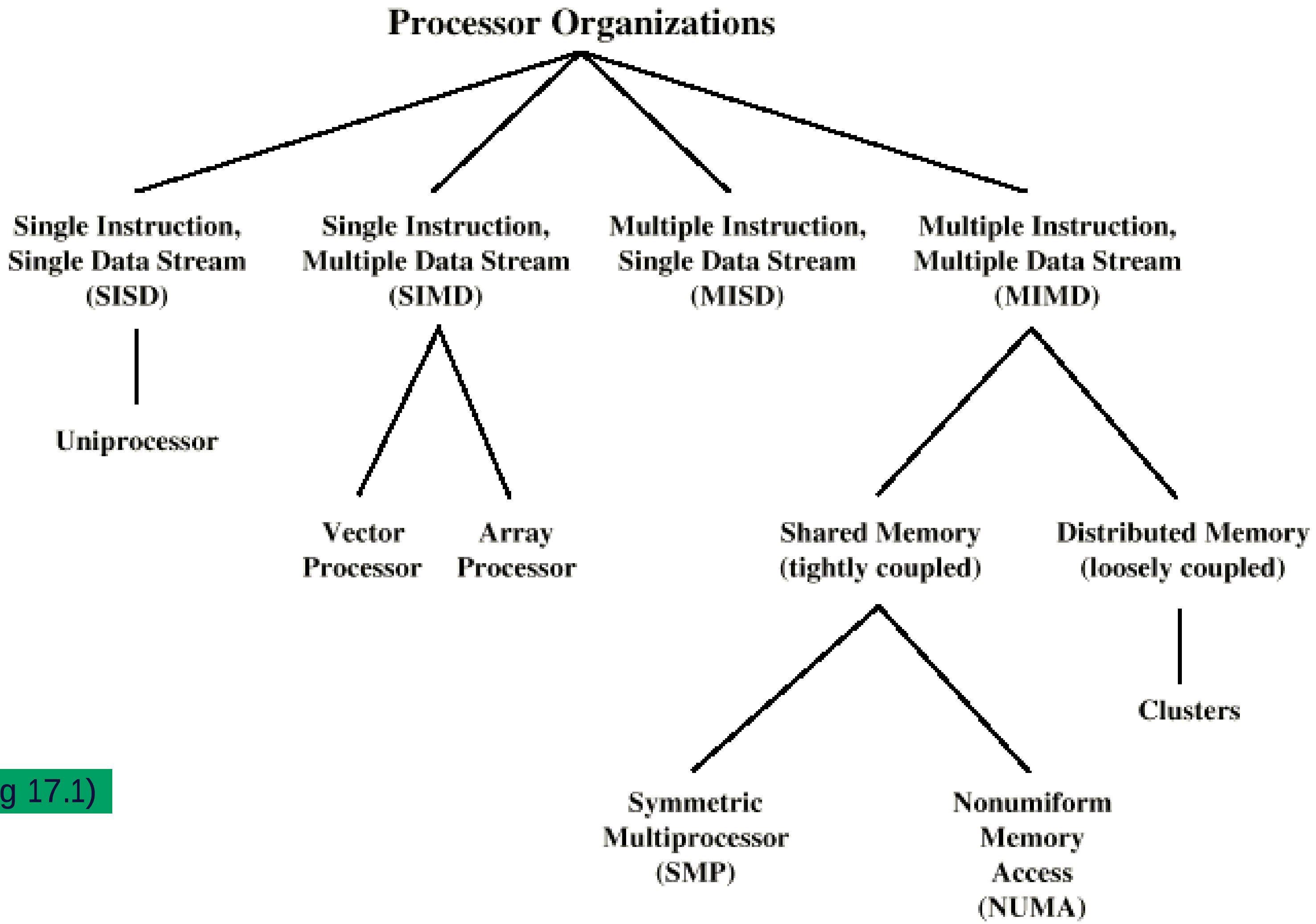
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Parallel Processing & Multicore computers

8th edition: Ch 17 & 18

Earlier editions contain only Parallel Processing

Parallel Processor Architectures
Flynn's taxonomy from 1972



Processor Organizations
Single Instruction, Single Data Stream (SISD)
Single Instruction, Multiple Data Stream (SIMD)
Multiple Instruction, Single Data Stream (MISD)
Multiple Instruction, Multiple Data Stream (MIMD)
Uniprocessor
Vector Processor
Array Processor
Shared Memory (tightly coupled)
Distributed Memory (loosely coupled)
Clusters
Symmetric Multiprocessor (SMP)
Nonumiform Memory Access (NUMA)

(Sta09 Fig 17.1)

Parallel Processor Architectures

- Single instruction, single data stream – **SISD**
  - Uniprocessor
- Single instruction, multiple data stream – **SIMD**
  - Vector and array processors
  - Single machine instruction controls simultaneous execution
  - Each instruction executed on different set of data by different processors
- Multiple instruction, single data stream – **MISD**
  - Sequence of data transmitted to set of processors
  - Each processor executes different instruction sequence
  - Not used
- Multiple instruction, multiple data stream- **MIMD**
  - Set of processors simultaneously execute different instruction sequences on different sets of data
  - SMPs, clusters and NUMA systems

Multiple instruction, multiple data stream- MIMD

- Differences in processor communication
- Symmetric Multiprocessor (SMP)
  - Tightly coupled – communication via shared memory
  - Share single memory or pool, shared bus to access memory
  - Memory **access time** of a given memory location is **approximately the same** for each processor
- Non-uniform memory access (NUMA)
  - Tightly coupled – communication via shared memory
  - **Access times** to different regions of memory may **differ**
- Clusters
  - Loosely coupled – no shared memory
  - Communication via fixed path or network connections
  - Collection of independent uniprocessors or SMPs

SMP – Symmetric Multiprocessor

- Two or more similar processors of comparable capacity
- All processors can perform the same functions (hence symmetric)
- Connected by a bus or other internal connection
- Share same memory and I/O
- I/O access to same devices through same or different channels
- Memory access time is approximately the same for each processor
- System controlled by integrated operating system
  - providing interaction between processors
  - Interaction at job, task, file and data element levels

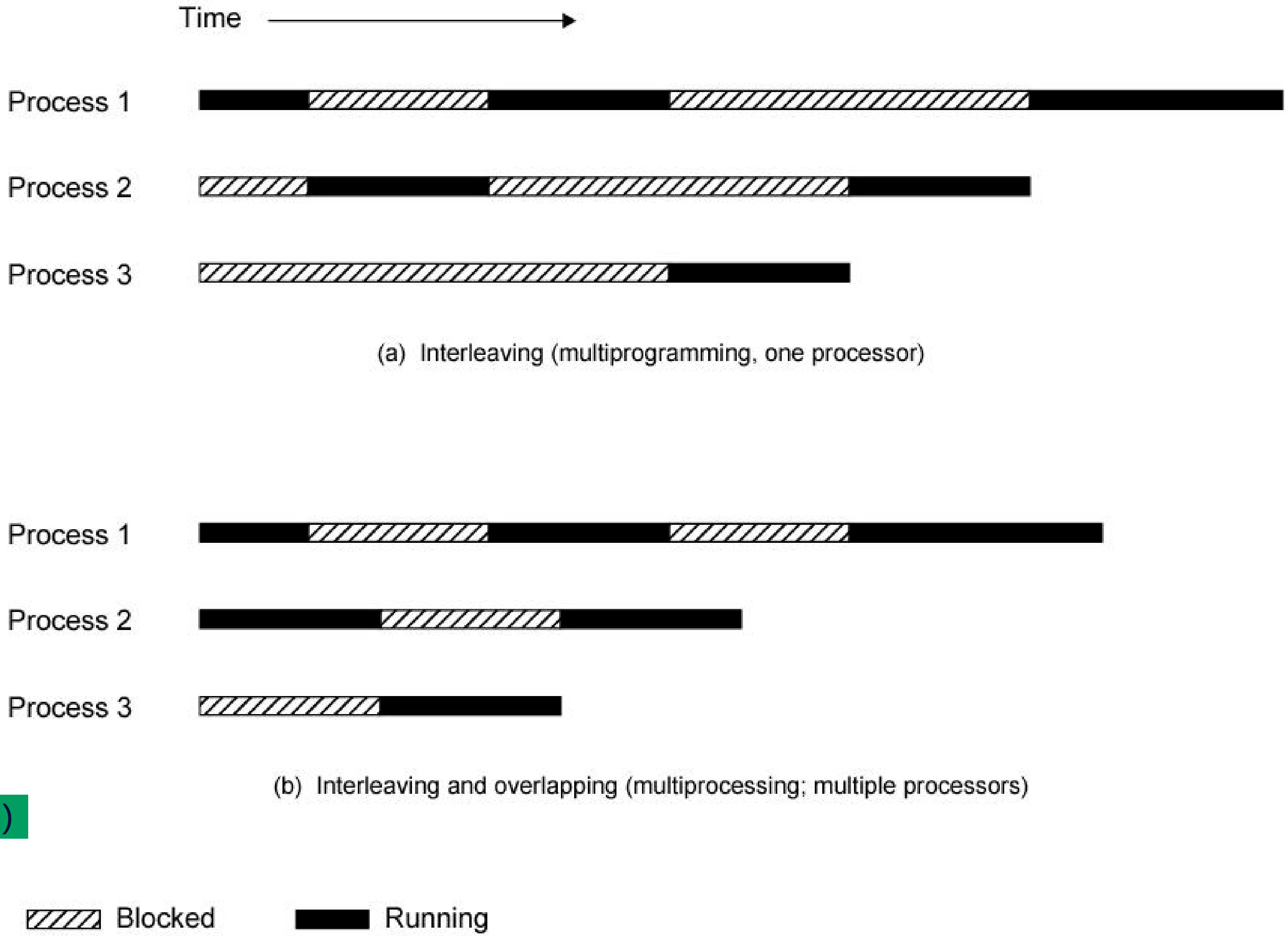
SMP – Advantages

- Performance
  - Only if some work can be done in parallel
- Availability
  - More processors to do the same functions
  - Failure of a single processor does not halt the system
- Incremental growth
  - Increase performance by adding additional processors
- Scaling
  - Different computers can have different number of processors
  - Vendors can offer range of products based on number of processors

Multiprogramming vs multiprocessing (*Moniajo*)

Multi-programming

Multi-processing



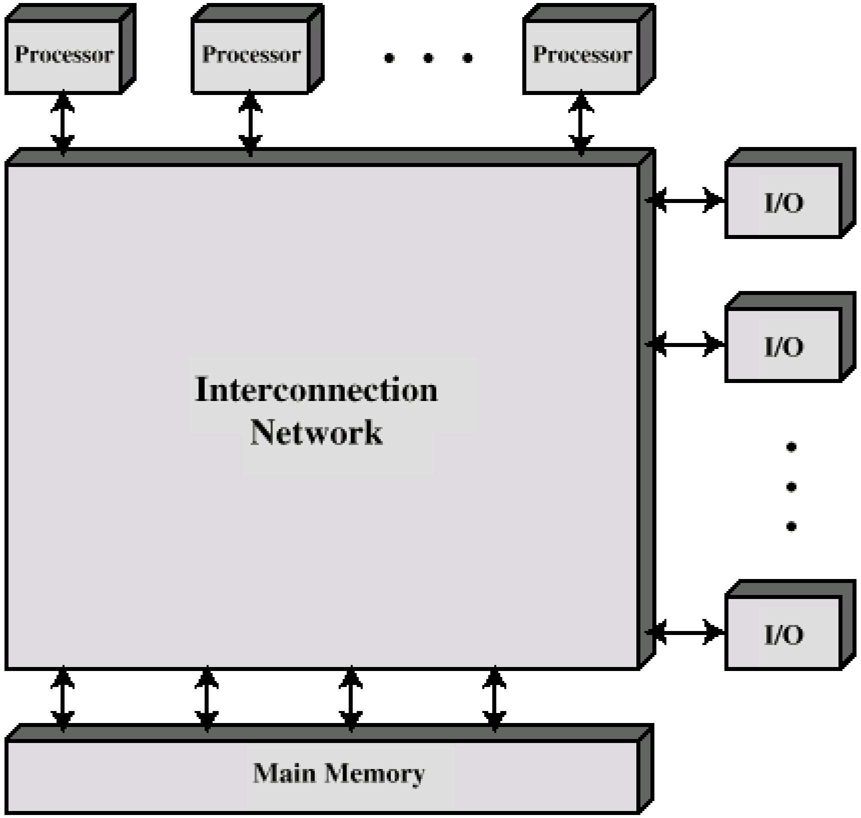
(a) Interleaving (multiprogramming, one processor)

(b) Interleaving and overlapping (multiprocessing; multiple processors)

(Sta09 Fig 17.3)

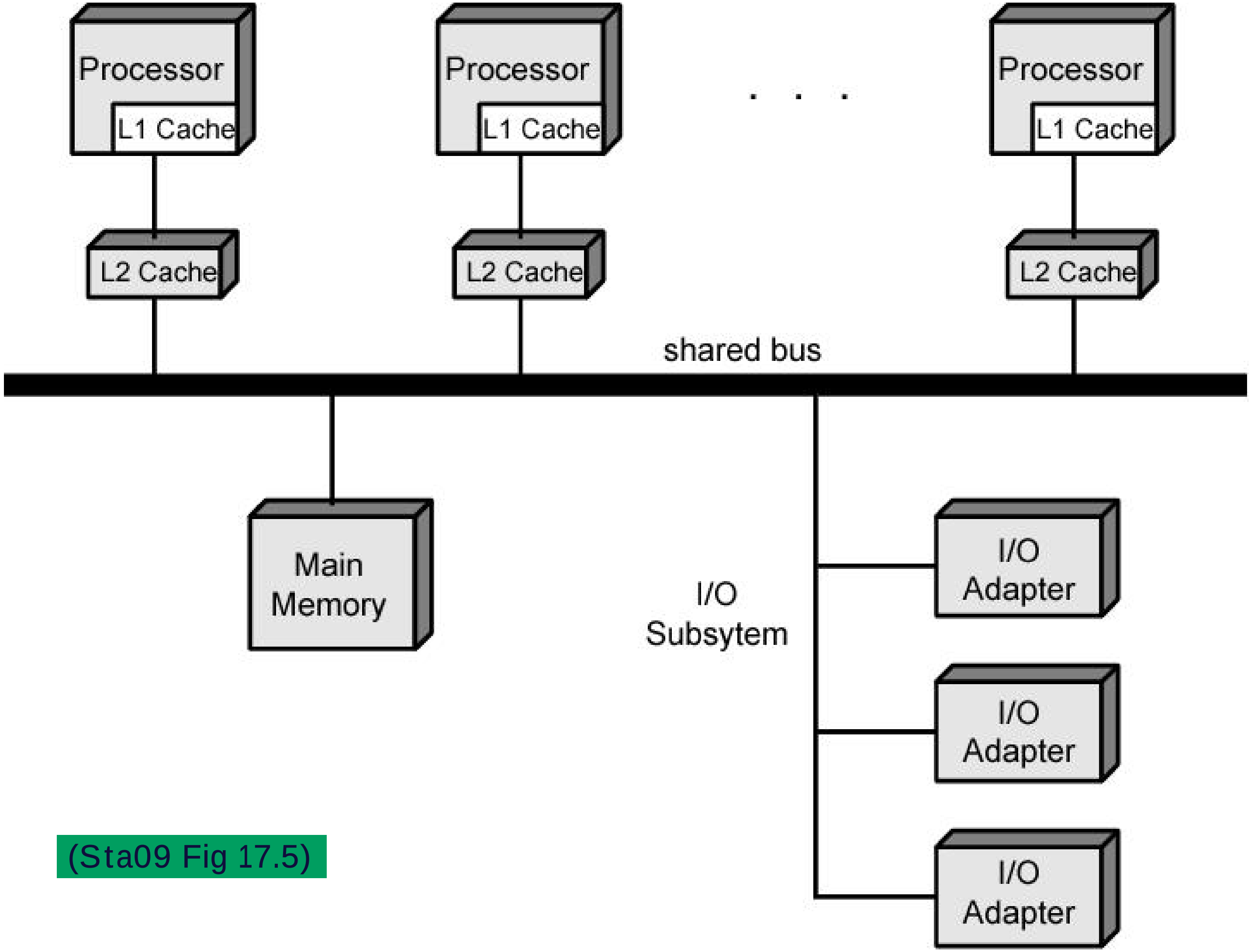
Multiprocessor Organization

- Processors
  - Two or more
  - Self-contained
  - Additionally, may have private memory and/or I/O channels
- Multiport memory
  - Shared memory
  - Simultaneous access to separate blocks
- Interconnection
  - Most common: Time shared bus



(Sta09 Fig 17.4)

Example: SMP Organization



(Sta09 Fig 17.5)

Time-shared bus

Advantages

- Simplicity
  - Addressing, arbitration and time-sharing logic same as in uniprocessor system
- Flexibility
  - Expand by attaching more processors to the bus
- Reliability
  - Bus is passive, failure of attached device should not cause failure of the whole

Disadvantages

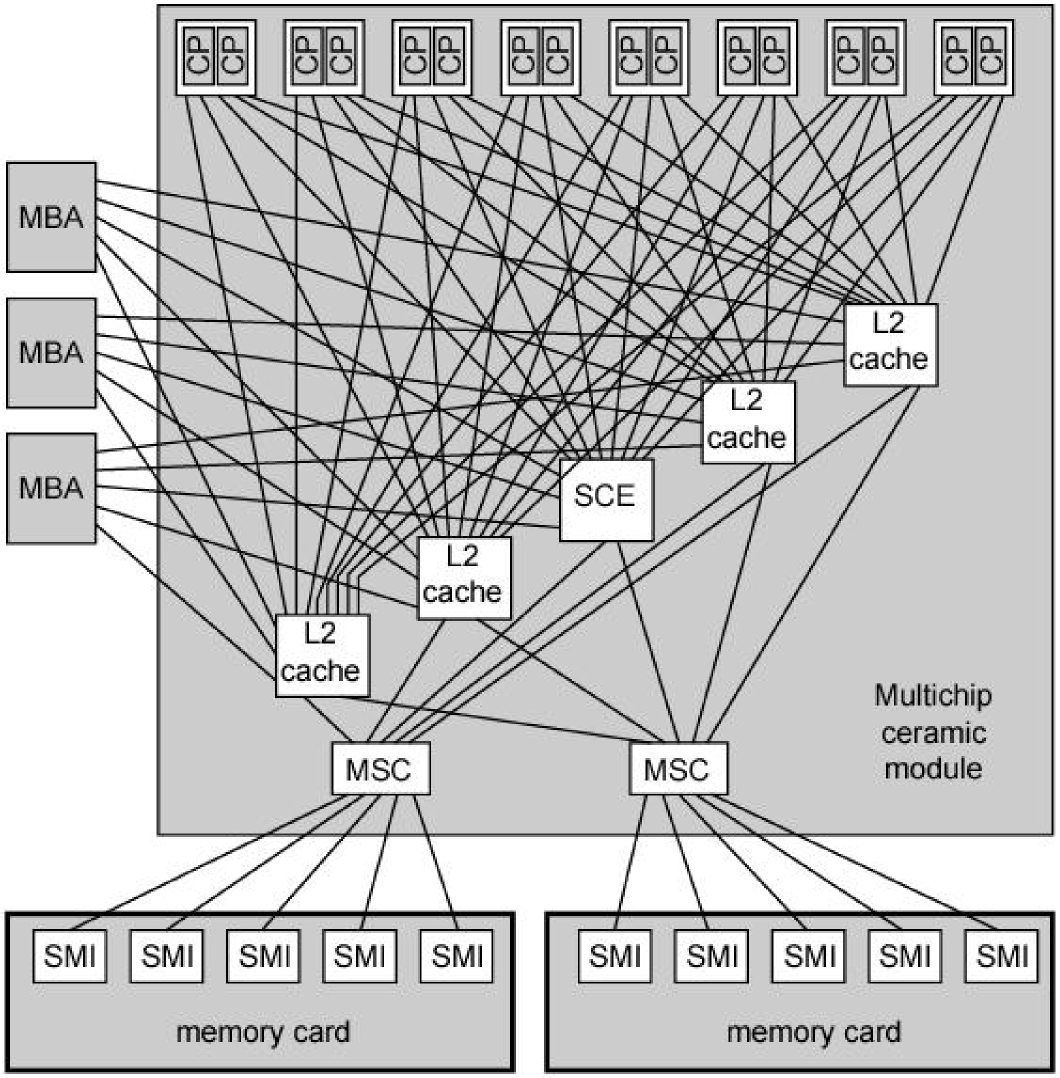
- Performance limited by bus cycle time
- Each processor should have local cache
  - Reduce number of bus accesses
- Leads to problems with <u>cache coherence</u>
  - Solved in hardware - see later

New requirements to operating system

- Simultaneous concurrent processes
  - Reentrant OS routines
  - OS data structure synchronization – avoid deadlocks etc.
- Scheduling
  - On SMP any processor may execute scheduler at any time
- Synchronization
  - Controlled access to shared resources
- Memory management
  - Use parallel access options
- Reliability and fault tolerance
  - Graceful degradation in the face of single processor failure

IBM z990 Multiprocessor Structure

- Dual-core processor chip
  - CISC superscalar
  - 256-kB L1 instruction and a 256-kB L1 data cache
  - Point-to-point conn. to L2
- L2 cache 32 MB
  - Clusters of five
  - Each cluster supports eight processors and access to entire main memory space
- System control element (SCE) (one of the L2 caches)
  - Arbitrates system comm.
  - Maintains cache coherence
- Memory card
  - Each 32 GB, Maximum 8
  - Interconnect to MSC via synchronous memory interfaces (SMIs)
- Memory bus adapter (MBA)
  - Interface to I/O channels, go directly to L2 cache



CP = central processor
MBA = memory bus adapter
MSC = main store control
SCE = system control element
SMI = synchronous memory interface

(Sta09 Fig 17.6)

Cache Coherence

(*välimuistin yhtenäisyys*)

Cache and data consistency

- Multiple processors with their own caches
  - Multiple copies of same data in different caches
  - Concurrent modification of the same data
- Could result in an inconsistent view of memory
  - Inconsistency – the values in caches are different
- Write back policy
  - Write first to local cache and only later to memory
- Write through policy
  - The value is written to memory when changed
  - Other caches must monitor memory traffic
- Solution: **maintain cache coherence**
  - Keep recently used variables in appropriate cache(s), while maintaining the consistency of shared variables!

Software solutions for coherence

- Compiler and operating system deal with problem
- Overhead transferred to compile time
- Design complexity transferred from hardware to software
- However, software tends to make conservative decisions
  - Inefficient cache utilization
- Analyze code to determine safe periods for caching shared variables

Hardware solutions for coherence

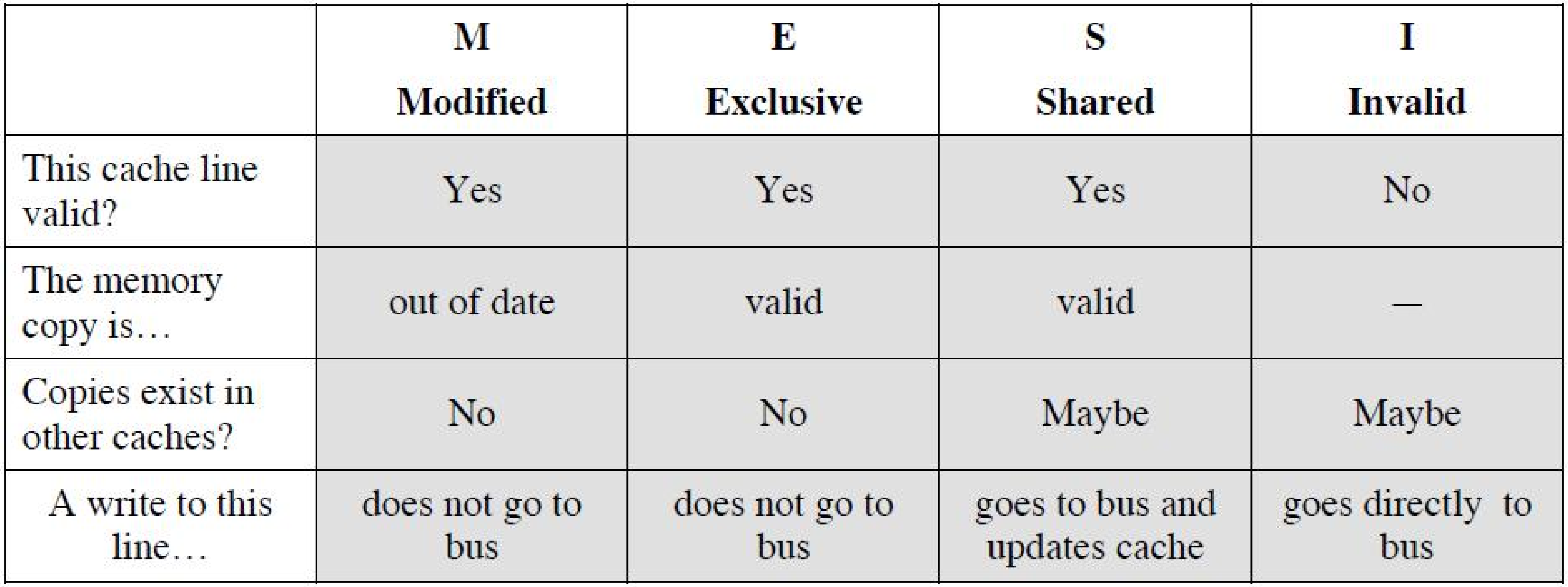
- Dynamic recognition of potential problems at run time
- More efficient use of cache, transparent to programmer
- Directory protocols
  - Collect and maintain information about copies of data in cache
  - Directory stored in main memory
  - Requests are checked against directory
  - Creates central bottleneck
  - Effective in large scale systems with complex interconnections
- Snoopy protocols
  - Distribute cache coherence responsibility to all cache controllers
  - Cache recognizes that a line is shared
  - Updates announced to other caches
  - Suited to bus based multiprocessor

Snoopy protocols: Write invalidate or update

- Write-Invalidate
  - Multiple readers, one writer
  - Write request invalidated that line in all other caches
  - Writing processor gains exclusive (cheap) access until line required by another processor
  - Used in Pentium II and PowerPC systems
  - State of every line marked as **m**odified, **e**xclusive, **s**hared or **i**nvalid (MESI)
- Write-Update
  - Multiple readers and writers
  - Updated word is distributed to all other processors
- Some systems use an adaptive mixture of both solutions

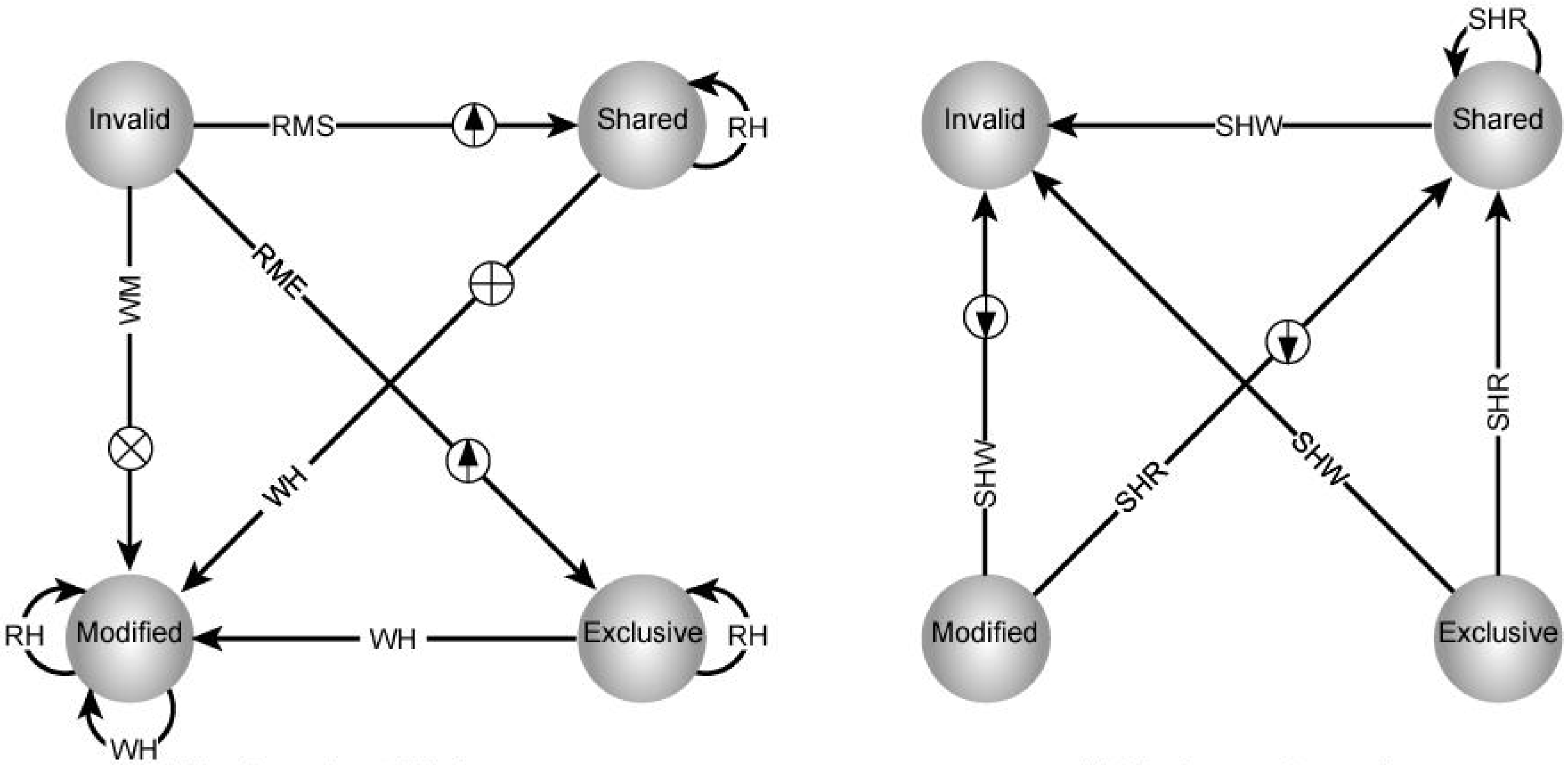
MESI Protocol - states

- Four states (two bits per tag)
  - Modified: modified (different than memory), only in this cache
  - Exlusive: only in this cache, but the same as memory
  - Shared: same as memory, may be other caches
  - Invalid: line does not contain valid data



	M Modified	E Exclusive	S Shared	I Invalid
This cache line valid?	Yes	Yes	Yes	No
The memory copy is…	out of date	valid	valid	—
Copies exist in other caches?	No	No	Maybe	Maybe
A write to this line…	does not go to bus	does not go to bus	goes to bus and updates cache	goes directly to bus

MESI State Transition Diagram



(a) Line in cache at initiating processor

(b) Line in snooping cache

RH	Read hit	⬇	Dirty line copyback
RMS	Read miss, shared	⊕	Invalidate transaction
RME	Read miss, exclusive	⊗	Read-with-intent-to-modify
WH	Write hit	⬆	Cache line fill
WM	Write miss		
SHR	Snoop hit on read		
SHW	Snoop hit on write or read-with-intent-to-modify		

(Sta09 Fig 17.6)

MESI Protocol – state transitions

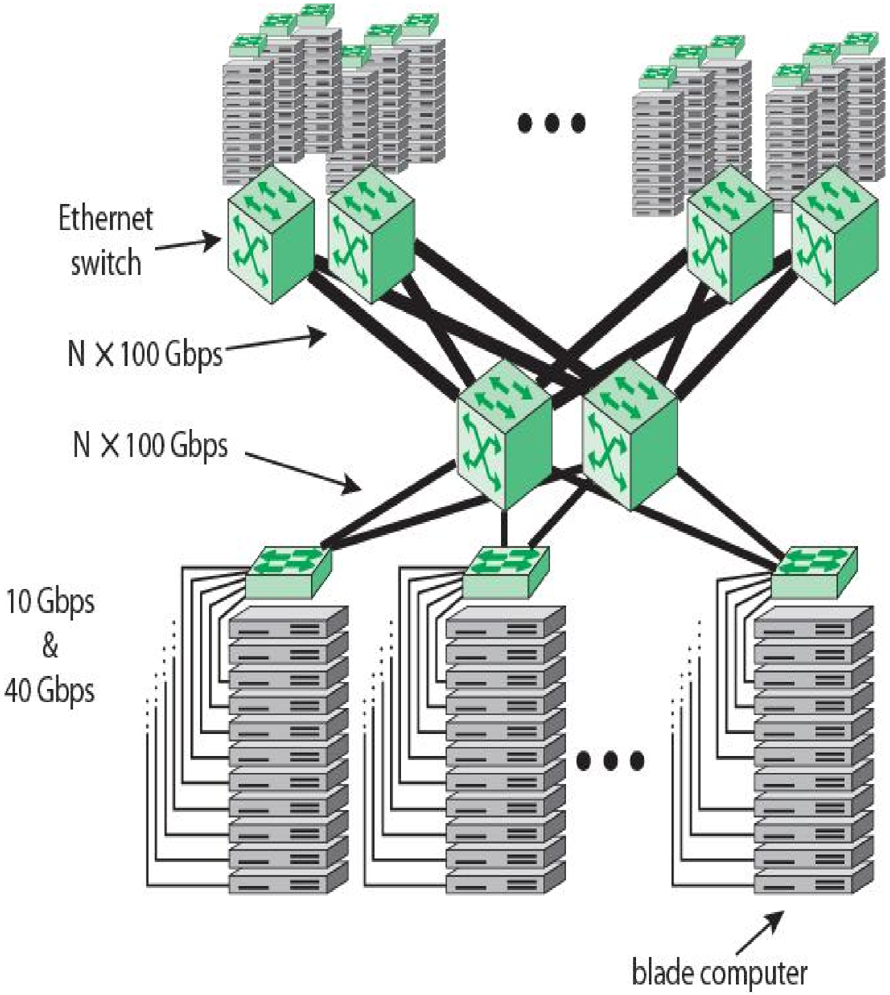
- Read Miss – generates SHR (snoop on read) to others
  - Not in any cache – simply read
  - Exclusive in some cache – SHR: exclusive ’owner’ indicates sharing and changes the state of its own cache line to shared
  - Shared in some caches – SHR: each signals about the sharing
  - Modified on some cache – SHR: memory read blocked, the content comes to memory and this cache from the other cache, which also changes the state of that line to shared
- Read Hit
- Write Miss – generates SHW (snoop on writes) to others
- Write Hit

Clusters

Cluster

(Sta09 Fig 17.12)

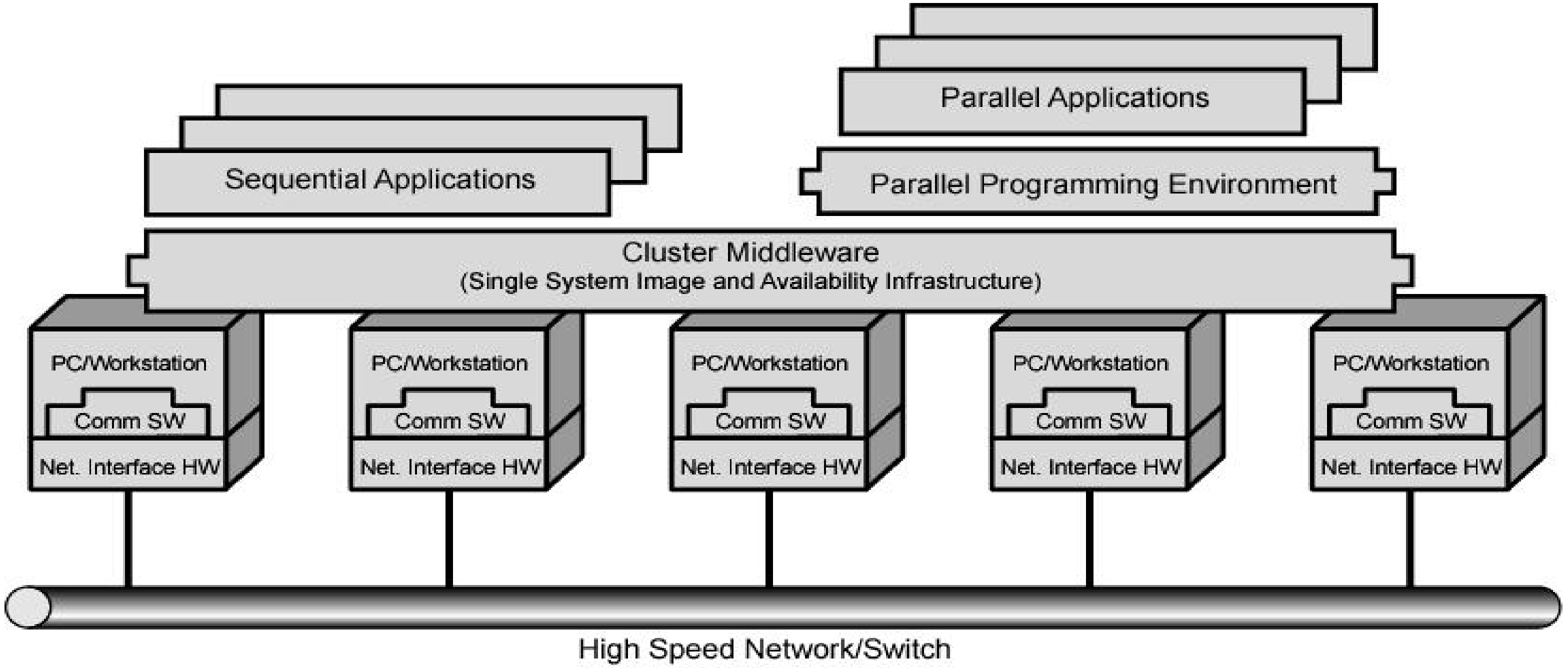
- Cluster is a group of interconnected nodes
  - Each node is a whole computer
  - Nodes work together as unified resource
  - Illusion of being one machine
  - Commonly for server applications
- Benefits:
  - Scalability
  - High availability
  - Load balancing
  - Superior price/performance



Example: blades in one or more chassis
blade – korttipalvelin, chassis – kehikko

Cluster Middleware

(Sta09 Fig 17.11)



- Unified image to user
  - Single system image
- Single point of entry
- Single file hierarchy
- Single control point
- Single virtual networking
- Single memory space
- Single job management system
- Single user interface
- Single I/O space
- Single process space
- Checkpointing
- Process migration

Department’s new research cluster (Not installed yet)

- 15 Chassis containing together 240 blades
  - Dell PowerEdge M1000e
  - 3 x 10 Gbit/s Dell PowerConnect M8024 for connections to other chassis and disk servers
- Each blade
  - Dell PowerEdge m610
  - 2 x Quad-core Xeon E5540 2,53 GHz
  - 32Gt RAM
  - 4 x 10 Gbit/s network connections
- Total 480 processors, 1920 simultaneus threads (SMT)
- One router and two switches to connect the blades together
- Going to use virtualization to form different configurations

NUMA –

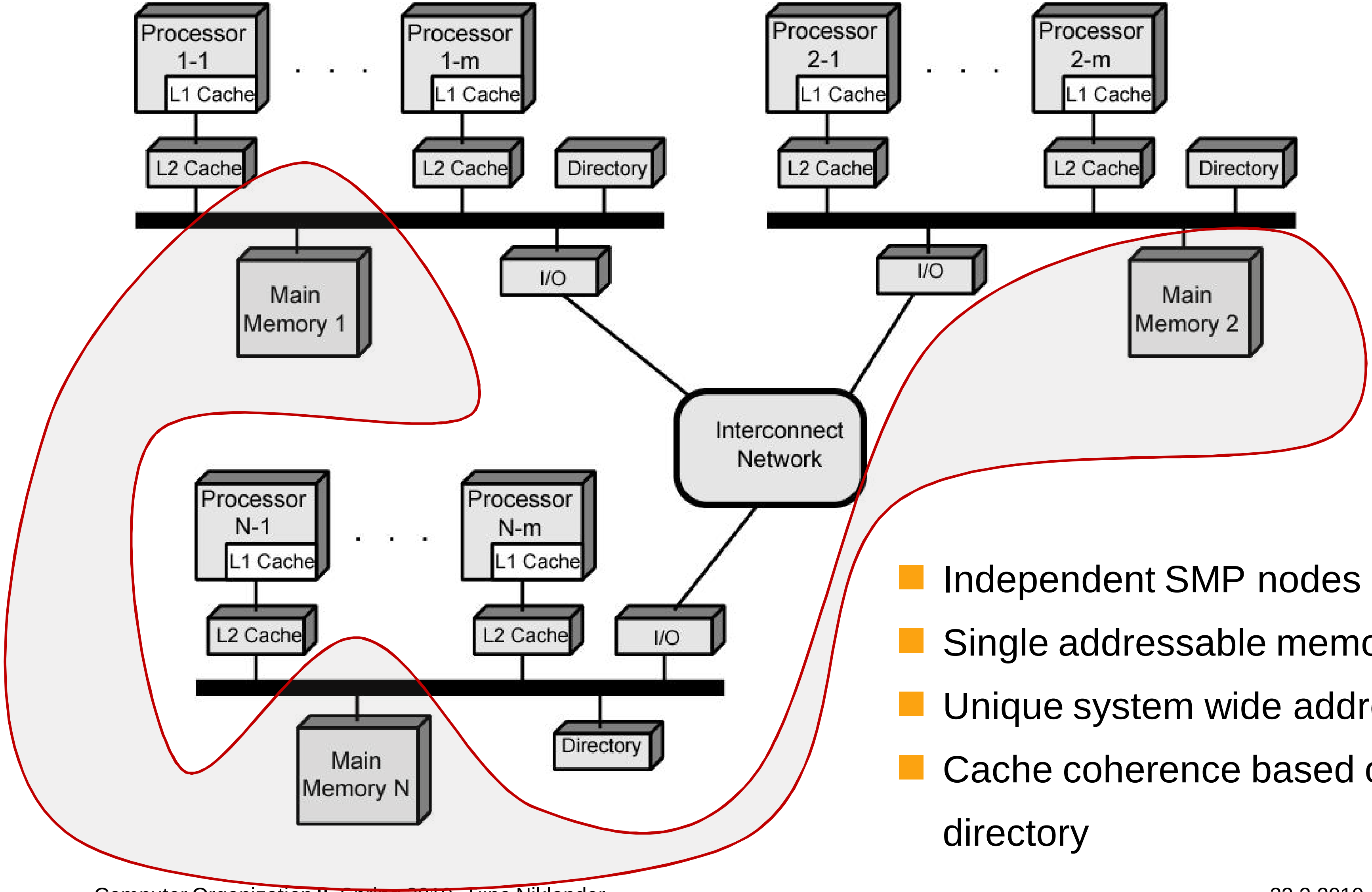
Nonuniform Memory Access

What is NUMA?

- SMP
  - Identical processors with uniform memory access (UMA) to shared memory
    - All processors can access all parts of the memory
    - Identical access time all memory regions for all processors
- Clusters
  - Interconnected computers with NO shared memory
- NUMA
  - All processors can access all parts of the memory
  - Access times to different regions are different for different processors
  - Cache-Coherent NUMA (CC-NUMA) maintains cache coherence among caches of various processors
  - **Maintain transparent system wide memory**

(Sta09 Fig 17.13)

CC-NUMA organization



- Independent SMP nodes
- Single addressable memory
- Unique system wide address
- Cache coherence based on directory

CC-NUMA – memory access

- Each processor has local L1 & L2 cache and main memory
- Nodes connected by some networking facility
- Each processor sees single addressable memory space
- Memory request order:
  - L1 cache (local to processor)
  - L2 cache (local to processor)
  - Main memory (local to node)
  - Remote memory (in other nodes)
    - Delivered to requesting (local to processor) cache
    - Needs to maintain cache coherence with other processor's caches
- Automatic and transparent

NUMA Pros & Cons

- Effective performance at higher levels of parallelism than SMP
- No major software changes
- Performance suffers if too much remote memory access
  - Avoid by good temporal and spatial locality of software with
    - L1 & L2 cache design to reduce all memory access
    - Virtual memory management move pages to nodes that use them most
- Not truly transparent memory access
  - Page allocation, process allocation and load balancing changes needed

- Shared-memory cluster?

Multicore computers

New chapter 18

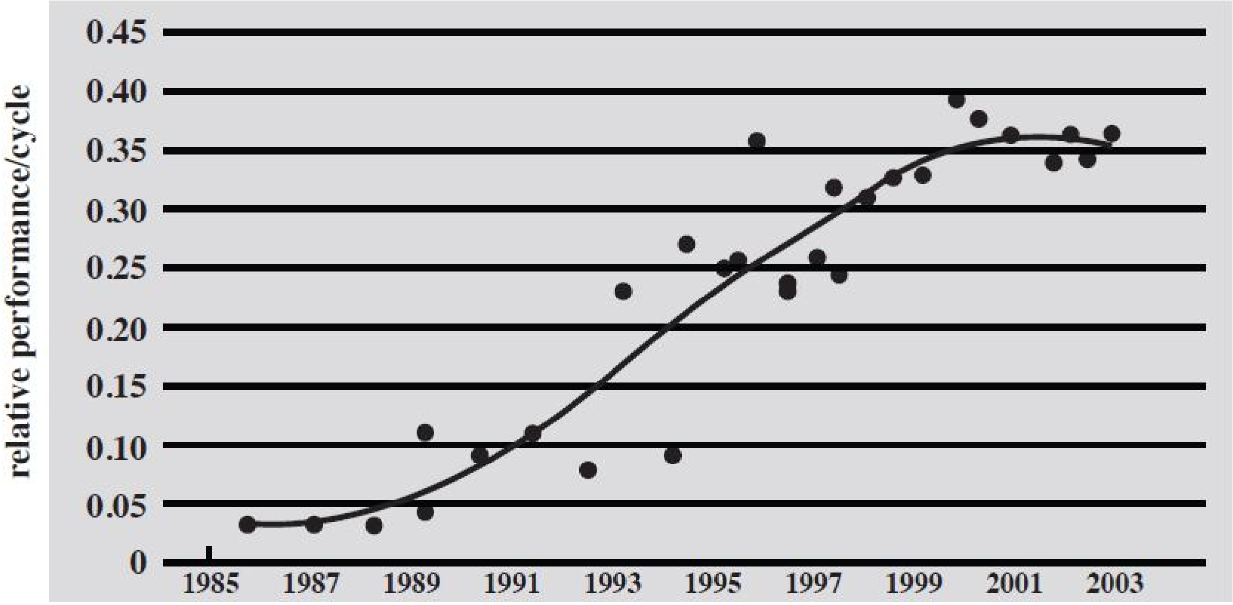
Why multicore?

- Current trend by processor manufacturers, because older improvements are no longer that promising
    - Clock frequency
    - Pipeline, superscalar,
    - Simultaneous multithreading , SMT (or hyperthreading)
- Enough transistors available on one chip to put two or more whole cores on the chip
    - Symmetric multiprocessor on one chip only
- But ... diminishing returns
    - More complexity requires more logic
    - Increasing chip area for coordinating and signal transfer logic
        - Harder to design, make and debug

Performance gains

(Sta09 Fig 18.2 b)

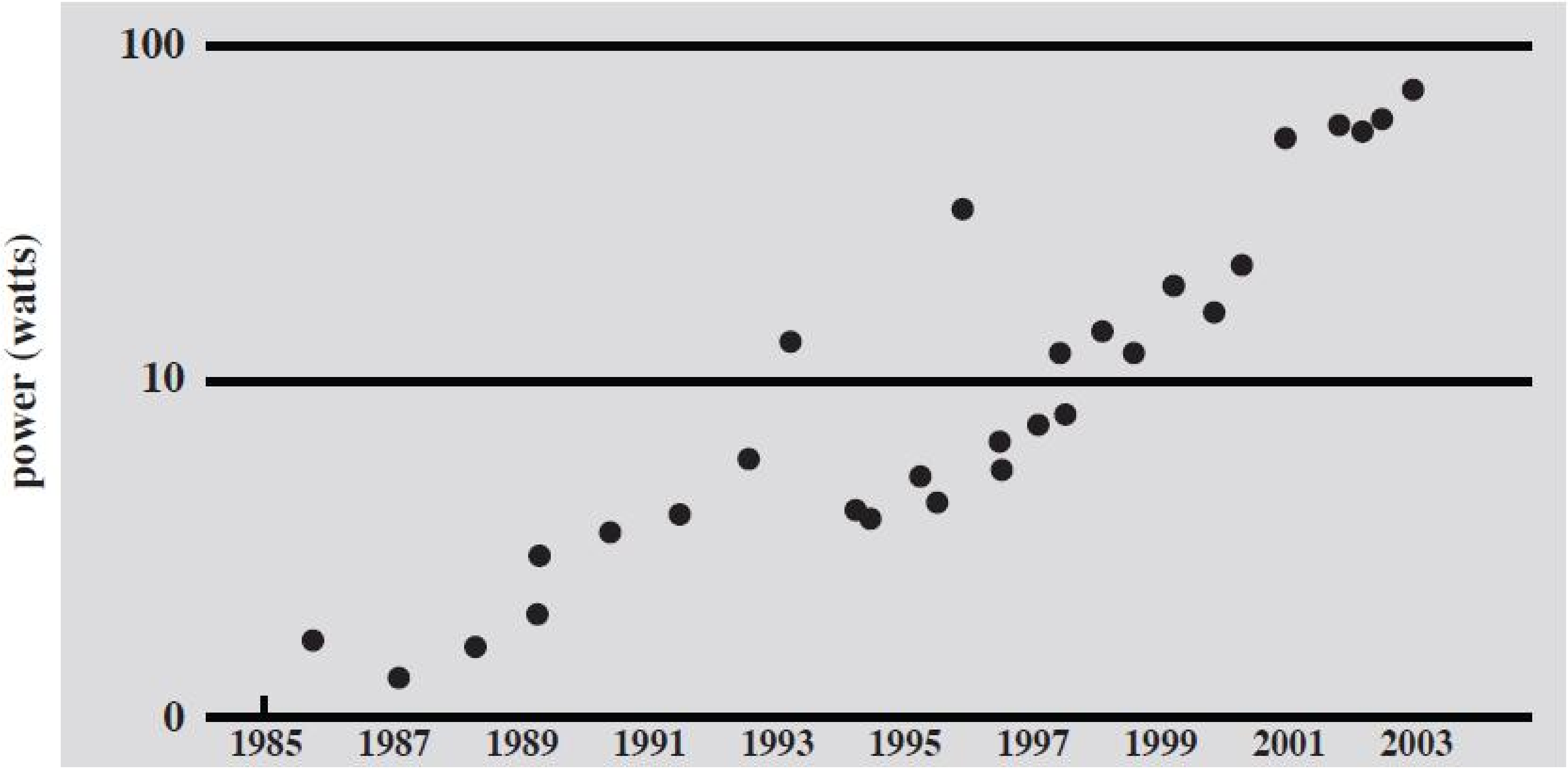
- Figure shows relative performance improvement in Intel provessors
- Dots calculated as a ratio of published SPEC CPU figures divided by clock frequency of that processor



- Late 1980's no parallelism yet – flat curve
- Steep rise of the curve with improvements in instruction-level parallelism
  - pipelines, superscalar, SMT
- Flat again around 2000 -> limit of instruction-level parallelism reached

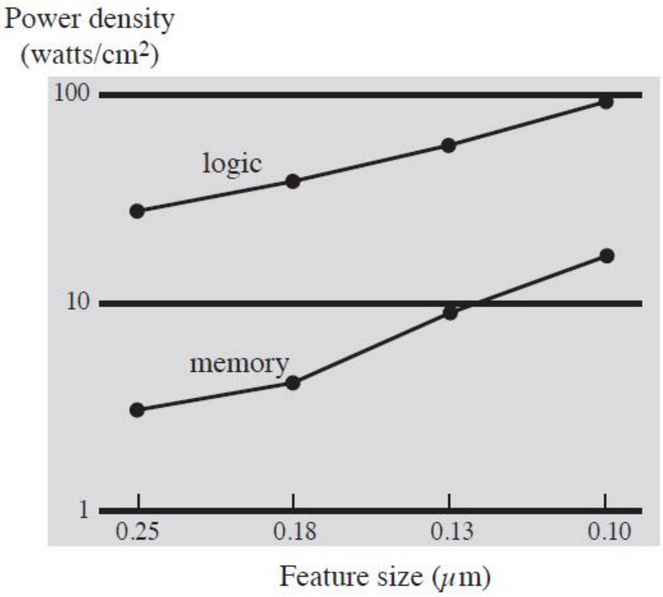
Power consumption

(Sta09 Fig 18.2 c)



- Power consumption of Intel processors
- Notice the prower requirement has grown exponentially

How to use all the transistors available?



(Sta09 Fig 18.3 a)

- Reduce power intensity by increasing the ratio of memory transistors to logic transistors
  - Memory transistors used mainly for cache
  - Logic transistors used for everything else
- Increased complexity in logic follows Pollack's rule
  - On a single core the increased complexity of structure means that more of the logic is needed just for coordination and signal transfer logic

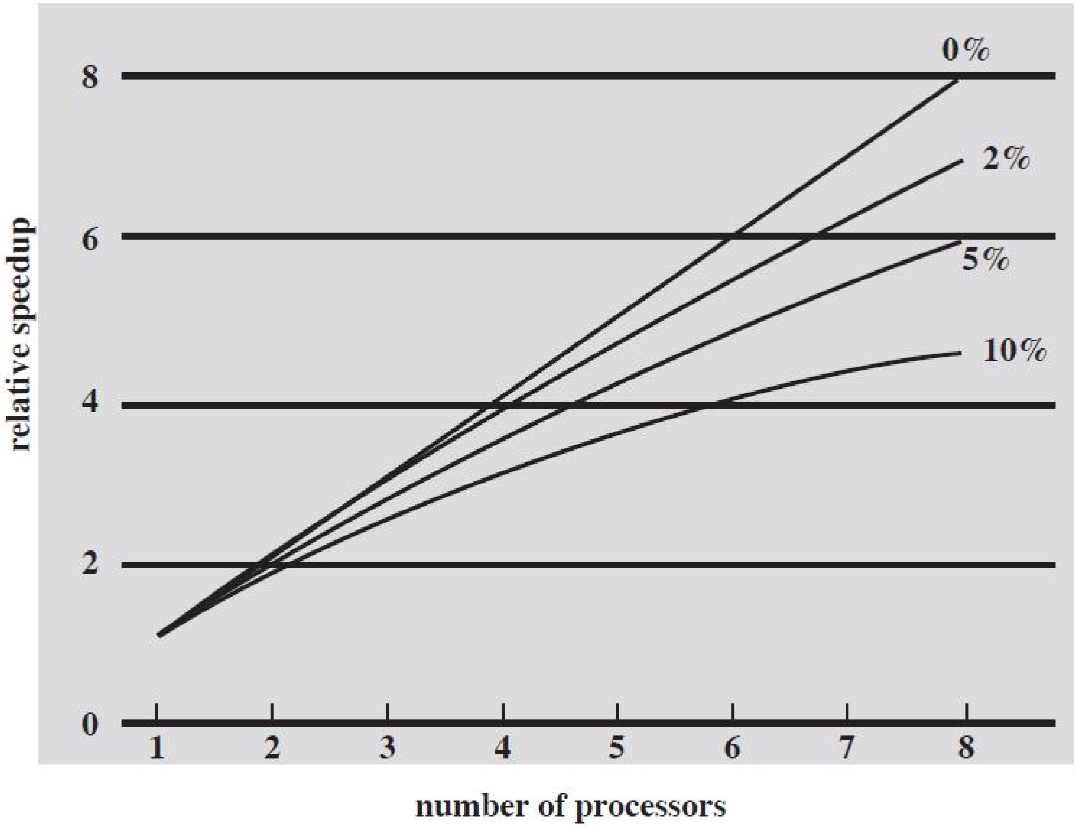
Pollack's rule

Performance increase is roughly proportional to [the] square root of [the] increase in complexity

Software performance on multicore

- Amdahl's law: speedup is proportional to the fraction of time enhancement is used
- Thus, even a small portion of sequential code has noticeable impact with larger number of processors!

- Software improvements not covered in this course

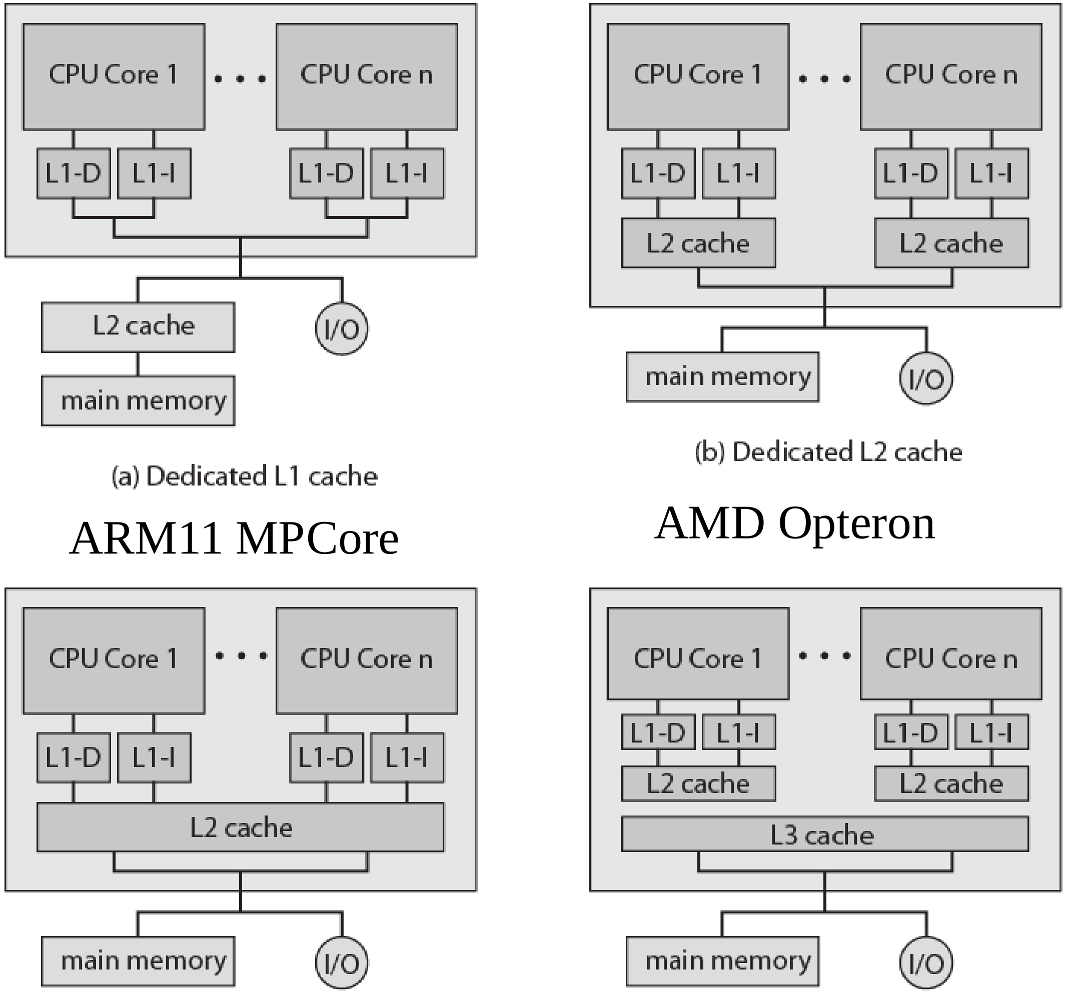


(a) Speedup with 0%, 2%, 5%, and 10% sequential portions

(Sta09 Fig 18.5 a)

Multicore organizations

(Sta09 Fig 18.8)



(a) Dedicated L1 cache

ARM11 MPCore

(b) Dedicated L2 cache

AMD Opteron

(c) Shared L2 cache

Intel Core Duo

(d) Shared L3 cache

Intel Core i7

- Key difference: Cache usage
- L1 always dedicated
  - Split for instructions and data
- L2 shared or dedicated (or mixed)
  - Active research on this issue
- L3 shared, if exists
- Remember cache coherence

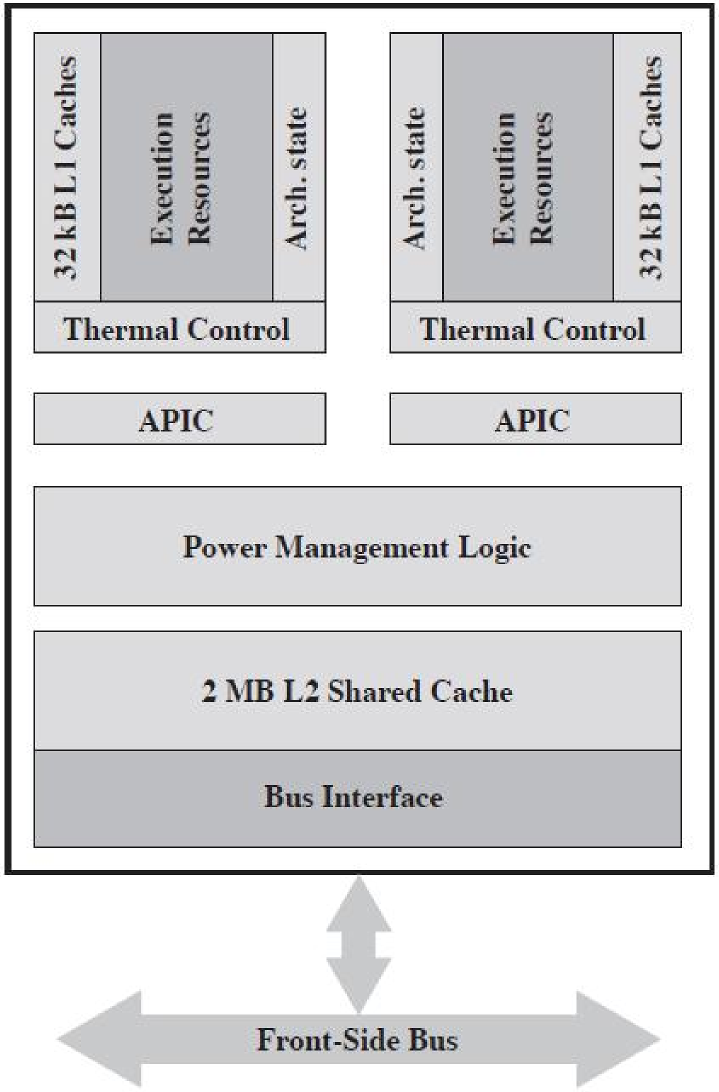
Shared L2 cache vs. dedicated ones

- Constructive interference
  - One core may fetch a cache line that is soon needed by another code – already available in shared cache
- Single copy
  - Shared data is not replicated, so there is just one copy of it.
- Dynamic allocation
  - The thread that has less locality needs more cache and may occupy more of the cache area
- Shared memory support
  - The shared data element already in the shared cache. With dedicated caches, the shared data must be invalidated from other caches before using
- Slower access
  - Larger cache area is slower to access, small dedicated cache would be faster

Intel Core Duo and Core i7

Intel Core Duo, 2006

(Sta09 Fig 18.9)

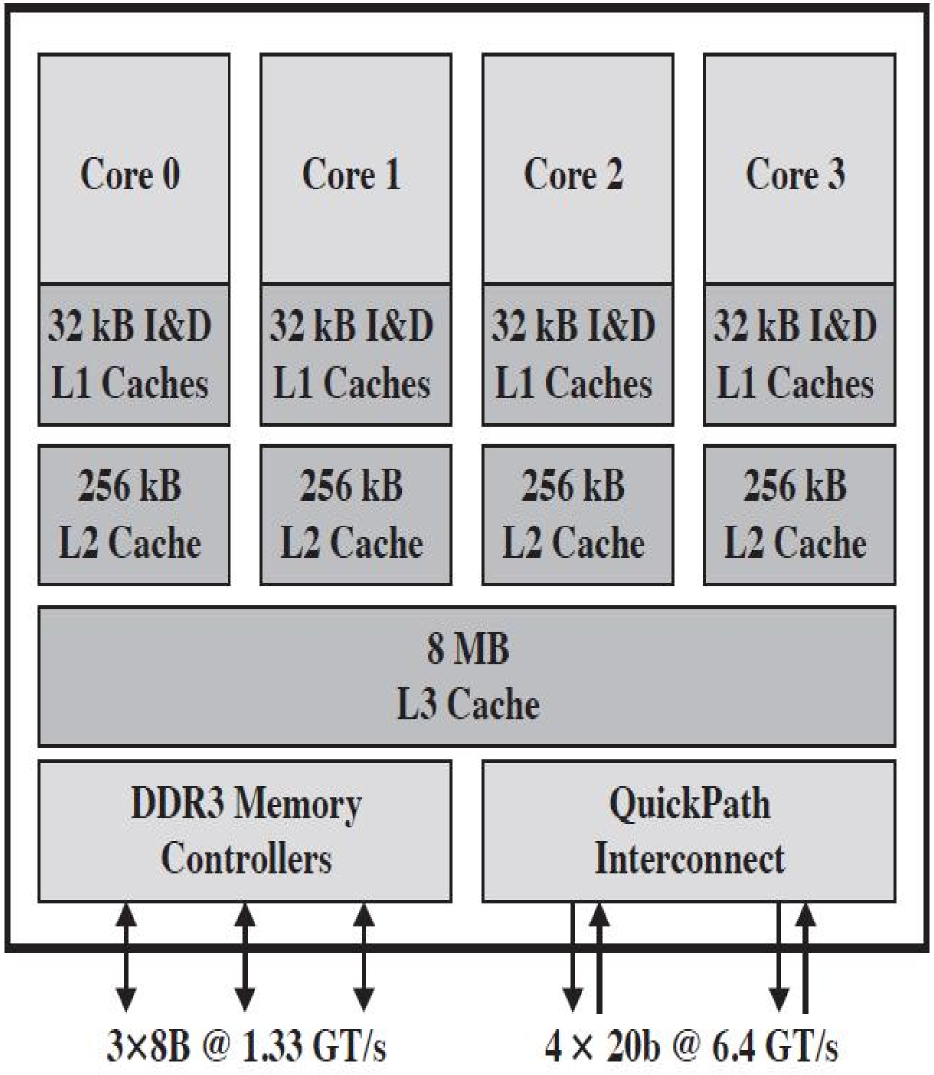


- Two x86 superscalar, shared L2 cache
  - MESI support for L1 caches
  - L2 data shared between local cores or external
- Thermal control unit per core
  - Manages chip heat dissipation
  - Maximize performance within constraints
- Advanced Programmable Interrupt Controlled (APIC)
  - Inter-process interrupts between cores
  - Routes interrupts to appropriate core
  - Includes timer so OS can interrupt core
- Power Management Logic
  - Adjusts voltage and power consumption
  - Can switch individual processor logic subsystems on and off

Intel Core i7 Block Diagram

(Sta09 Fig 18.10)

- Four x86 SMT processors each with two simultaneous threads
- Dedicated L2, shared L3 cache
- Speculative pre-fetch for caches
- On chip DDR3 memory controller
  - Three 8 byte channels (192 bits) giving 32GB/s
  - No front side bus
- QuickPath Interconnection
  - Cache coherent point-to-point link
  - High speed communications between processor chips
  - 6.4G transfers per second, 16 bits per transfer
  - Dedicated bi-directional pairs
  - Total bandwidth 25.6GB/s

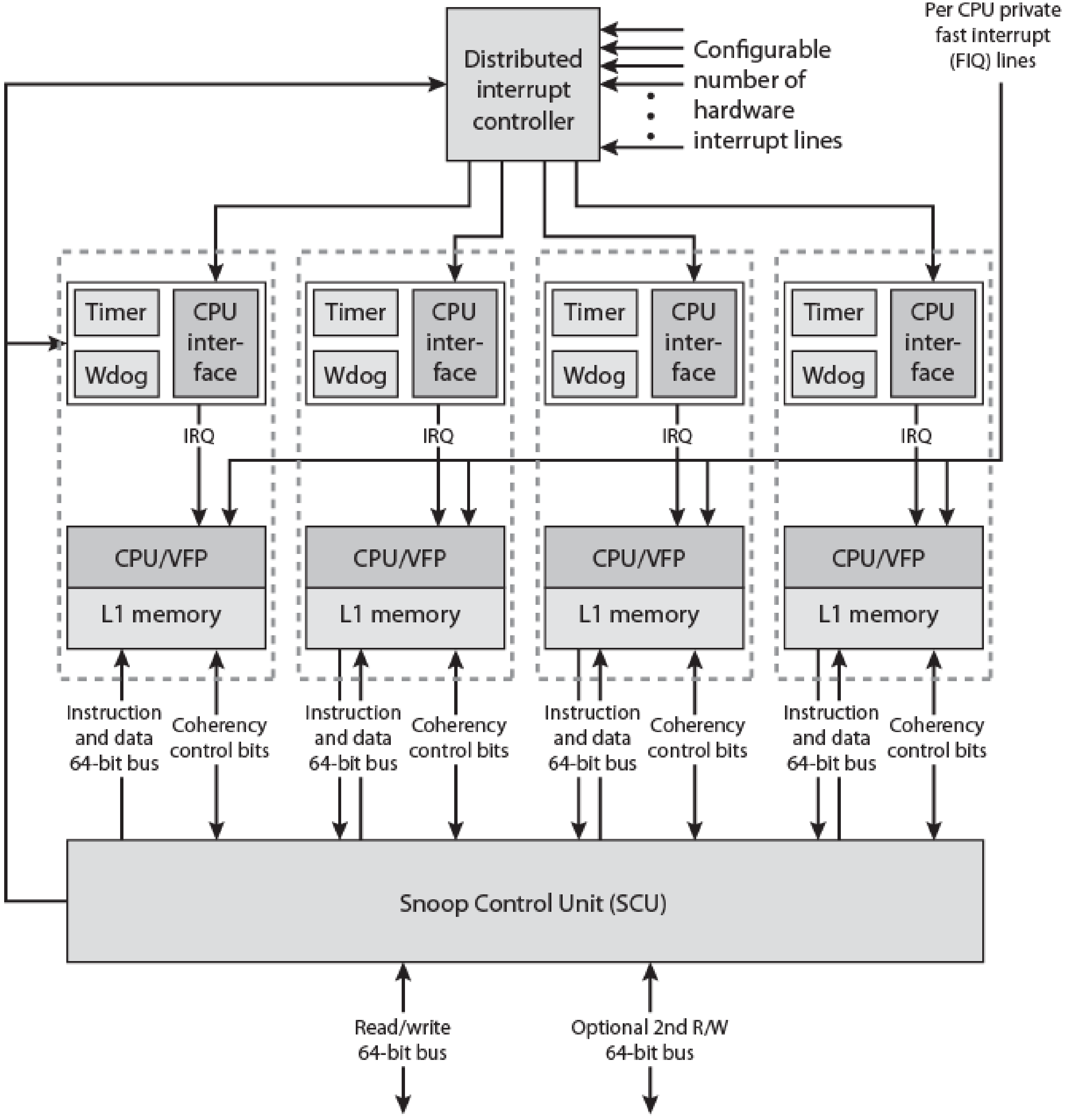


ARM11 MPCore

ARM11 MPCore

- Up to 4 processors each with own L1 instruction and data cache
- Distributed interrupt controller
- Timer per CPU
- Watchdog
  - Warning alerts for software failures
  - Counts down from predetermined values, issues warning at zero
- CPU interface
  - Interrupt acknowledgement, masking and completion acknowledgement
- CPU – Single ARM11 called MP11
- Vector floating-point unit
  - FP co-processor
- L1 cache
- Snoop control unit
  - L1 cache coherency

ARM11 MPCore



Distributed interrupt controller
Configurable number of hardware interrupt lines
Per CPU private fast interrupt (FIQ) lines
Timer
Wdog
CPU inter-face
IRQ
CPU/VFP
L1 memory
Instruction and data 64-bit bus
Coherency control bits
Snoop Control Unit (SCU)
Read/write 64-bit bus
Optional 2nd R/W 64-bit bus

(Sta09 Fig 18.11)

Interrupt Control

- Distributed Interrupt Controller (DIC)
  - collates interrupts from many sources
  - Masking, prioritization
  - Distribution to target MP11 CPUs
  - Status tracking (Interrupt states: pending, active, inactive)
  - Software interrupt generation
- Number of interrupts independent of MP11 CPU design
- Accessed by CPUs via private interface through SCU
- Can route interrupts to single or multiple CPUs
  - OS can generate interrupts: all-but-self, self, or specific CPU
- Provides inter-process communication (16 intr. ids)
  - Thread on one CPU can cause activity by thread on another CPU

Review Questions / Kertauskysymyksiä

- Cache coherence and MESI protocol

- Välimuistin yhtenäisyys (eheys) ja MESI protokolla