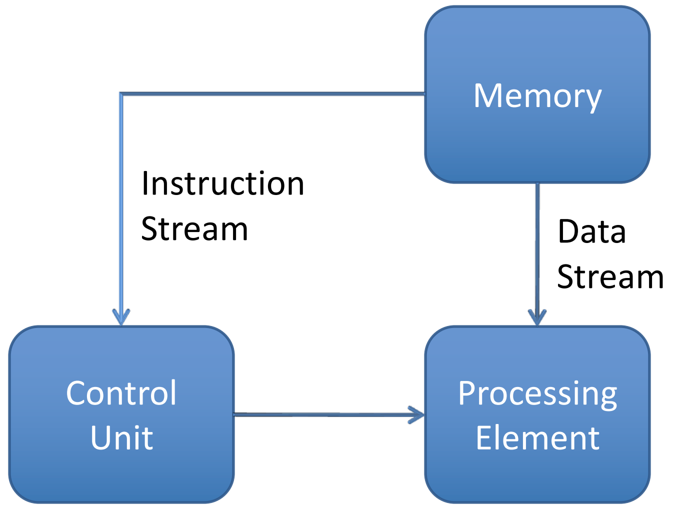
CS4223
Many- and multi-core for TLP

Trevor E. Carlson
National University of Singapore
tcarlson@comp.nus.edu.sg
(Slides from Tulika Mitra)

Flynn’s Taxonomy of Parallel Computers

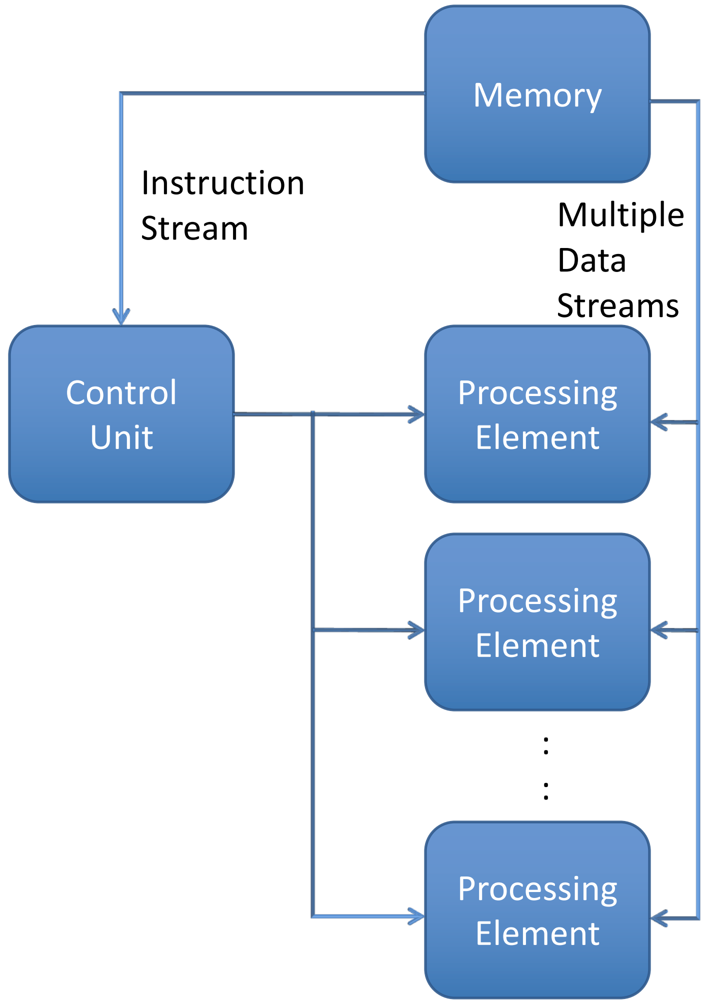
		Number of Data Streams	
		Single	Multiple
Number of instruction streams	Single	SISD	SIMD
	Multiple	MISD	MIMD

SISD: Single Instruction Single Data



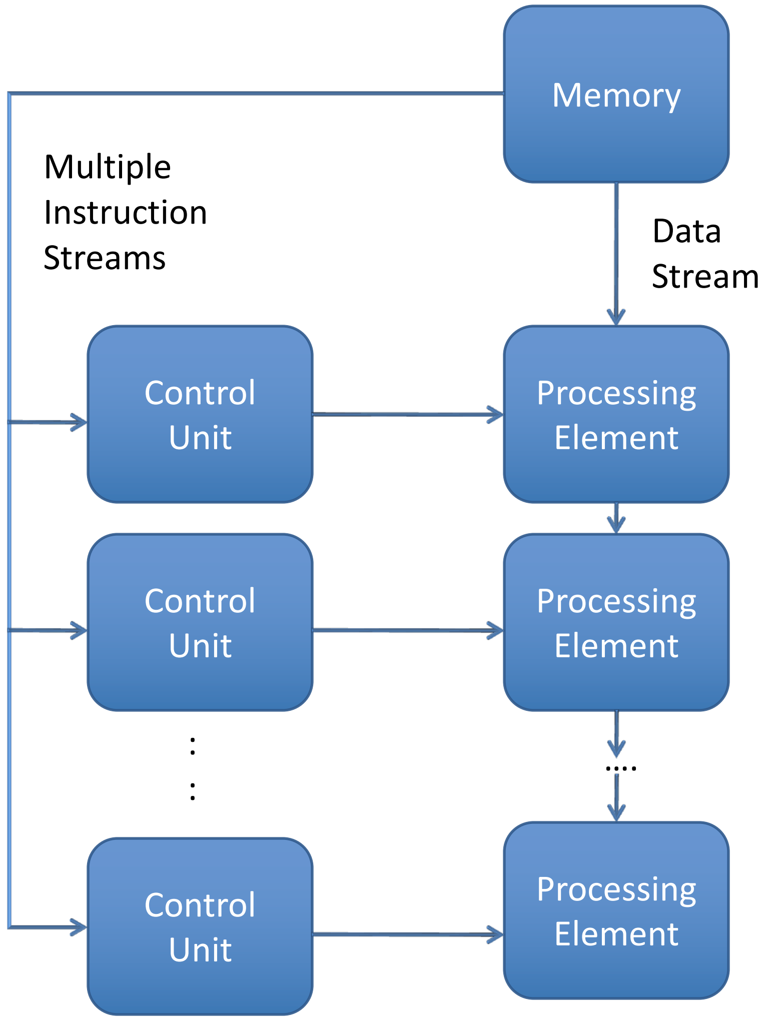
- SISD is not considered as a parallel architecture
- SISD exploits parallelism at the instruction level
- Pipelined, Superscalar, and VLIW architectures are examples of SISD architecture

SIMD: Single Instruction Multiple Data



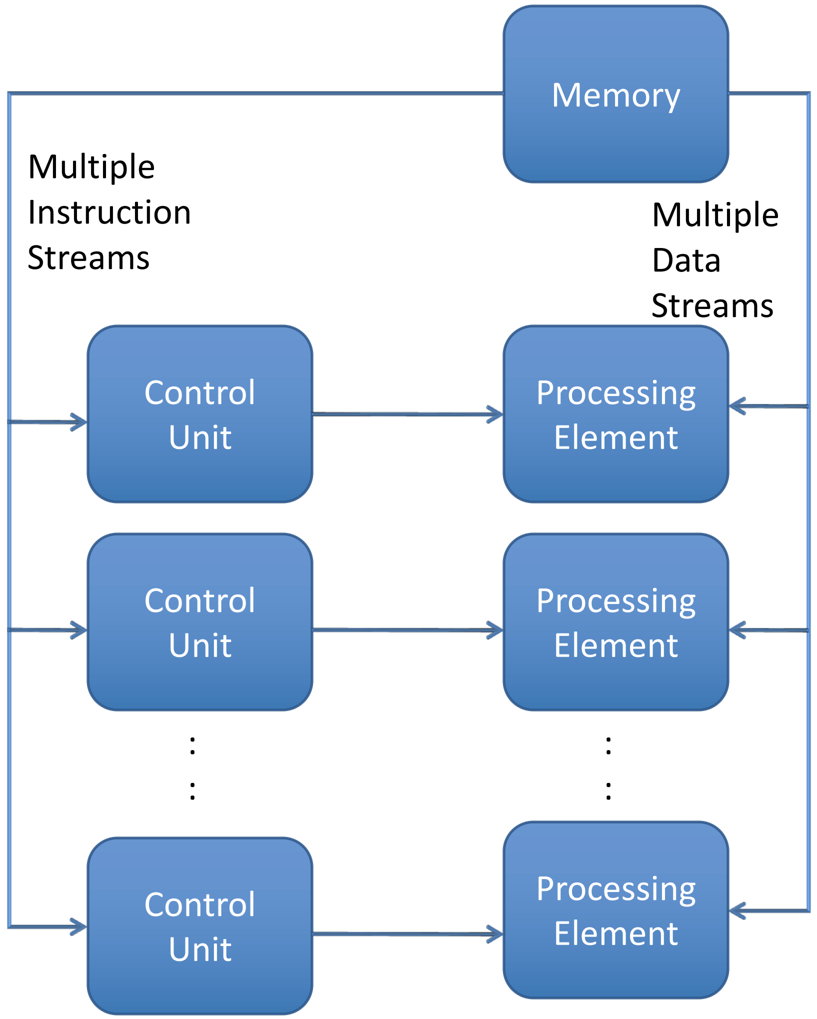
- A single instruction operates on multiple data to exploit data parallelism
- Vector processors and GPUs are excellent examples of SIMD architecture
- More efficient in terms of instruction count and loop control overhead

MISD: Multiple Instruction Single Data



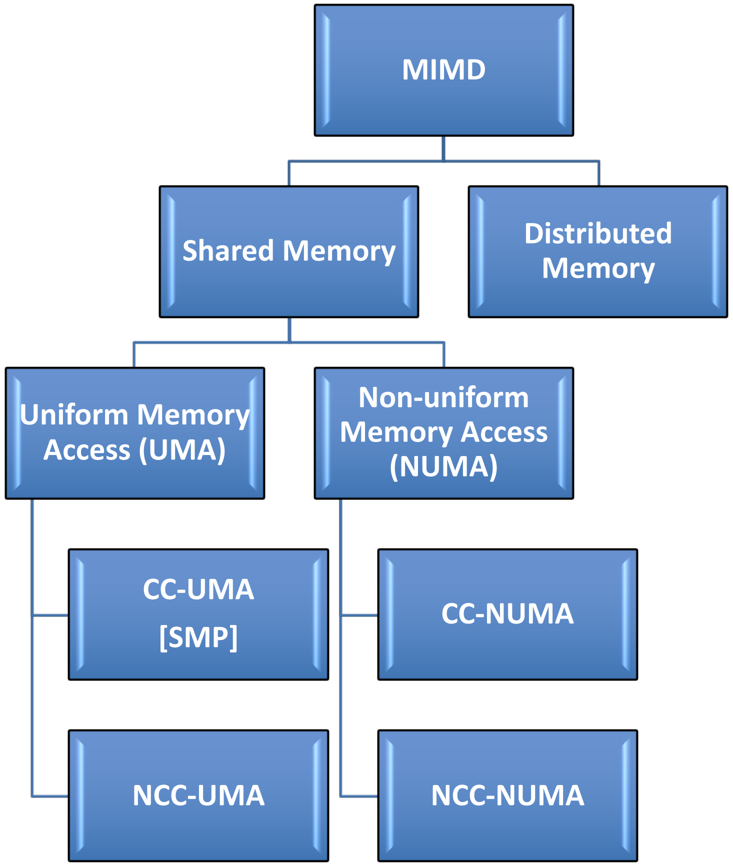
- Multiple processing elements execute from different instruction streams and data is passed from one processing element to the next
- Example: Systolic array such as CMU iWrap
- Data passing restriction is quite severe --- hard to generalize

MIMD: Multiple Instruction Multiple Data



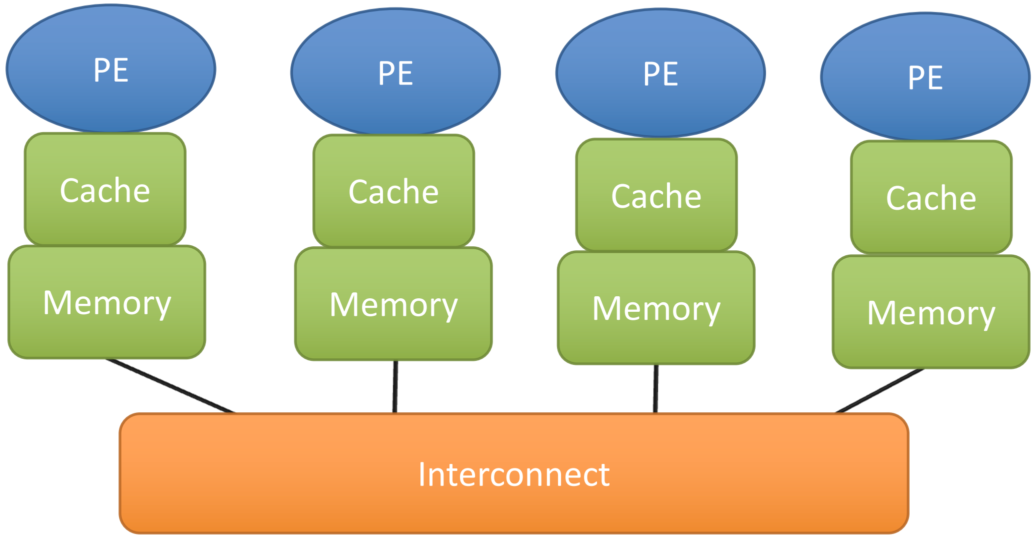
- Most flexible architecture
- Used in most parallel computers today

MIMD Classification



Distributed Memory MIMD

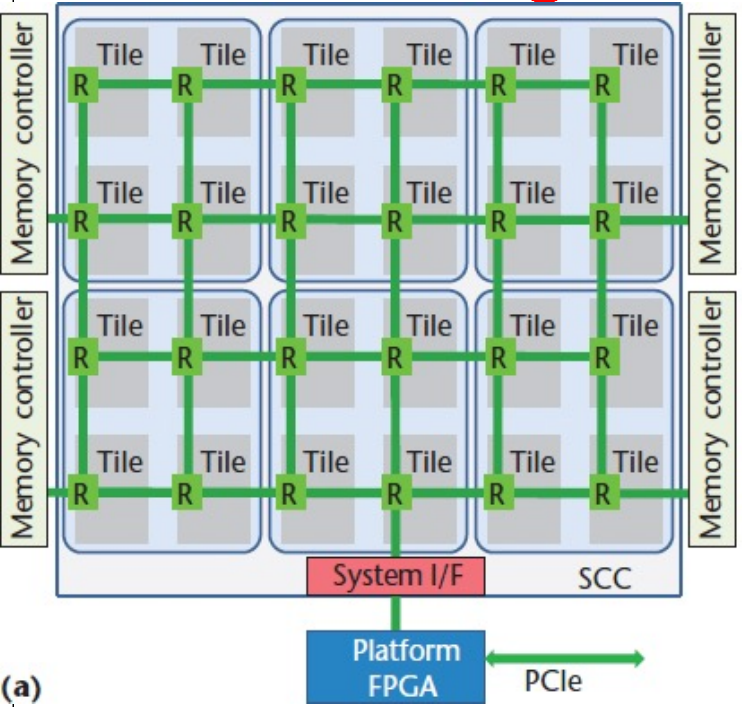
- Processing elements (PE) work independently
- Interaction is via message passing
- PEs cannot access the memory of another PE directly

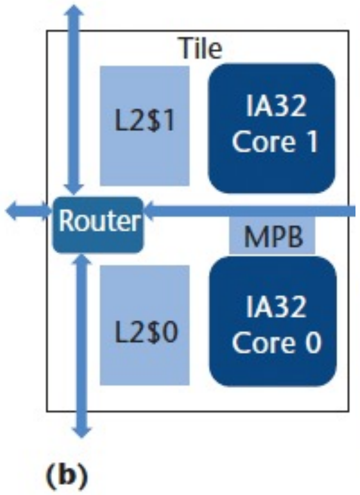


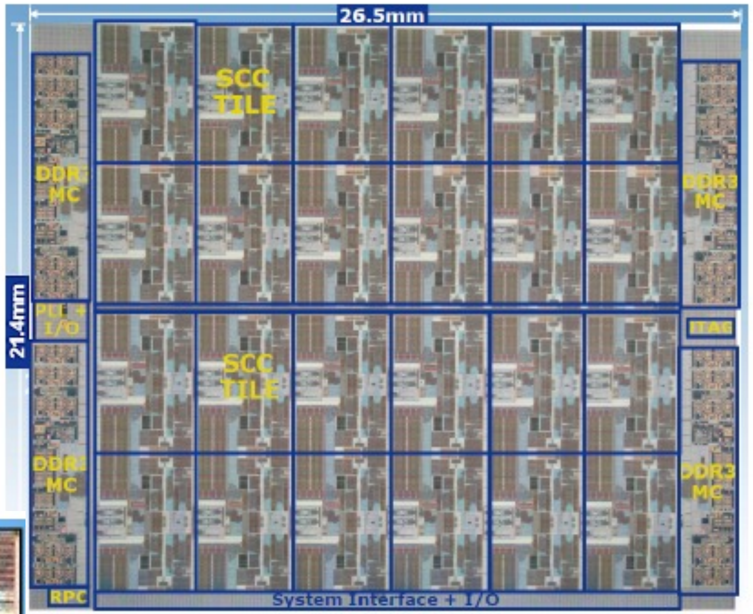
Distributed Memory MIMD (contd.)

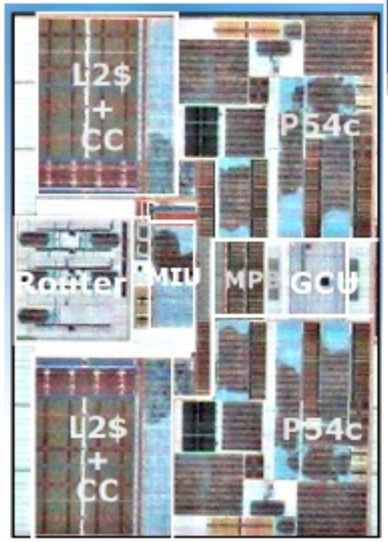
- Advantages
  - Processors mostly work with local memory
    - less contention and highly scalable
  - As communication is via message passing, sophisticated synchronization techniques are not required
- Disadvantages
  - Load balancing (partition of code and data) responsibility is on the programmer
  - Message-passing-based synchronization can lead to deadlocks; programmer's responsibility
  - Performance overhead due to physical data copy

Intel Single Chip Cloud Computer (SCC)



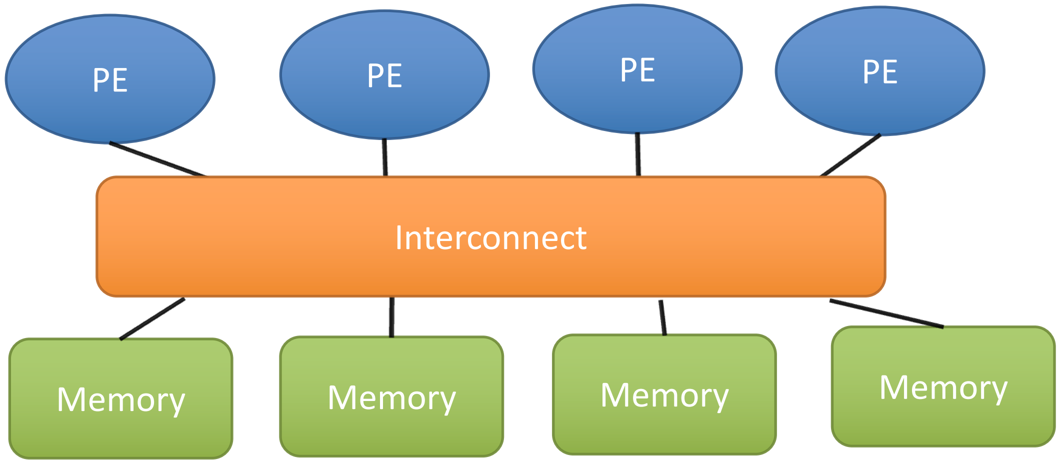






Shared Memory MIMD

- Multiple processors with a logically shared memory
- The memory modules define a global address space that is shared among the processors
- Any processor can access any memory module
- Communication though memory loads and stores
- Logically shared memory can be physically distributed

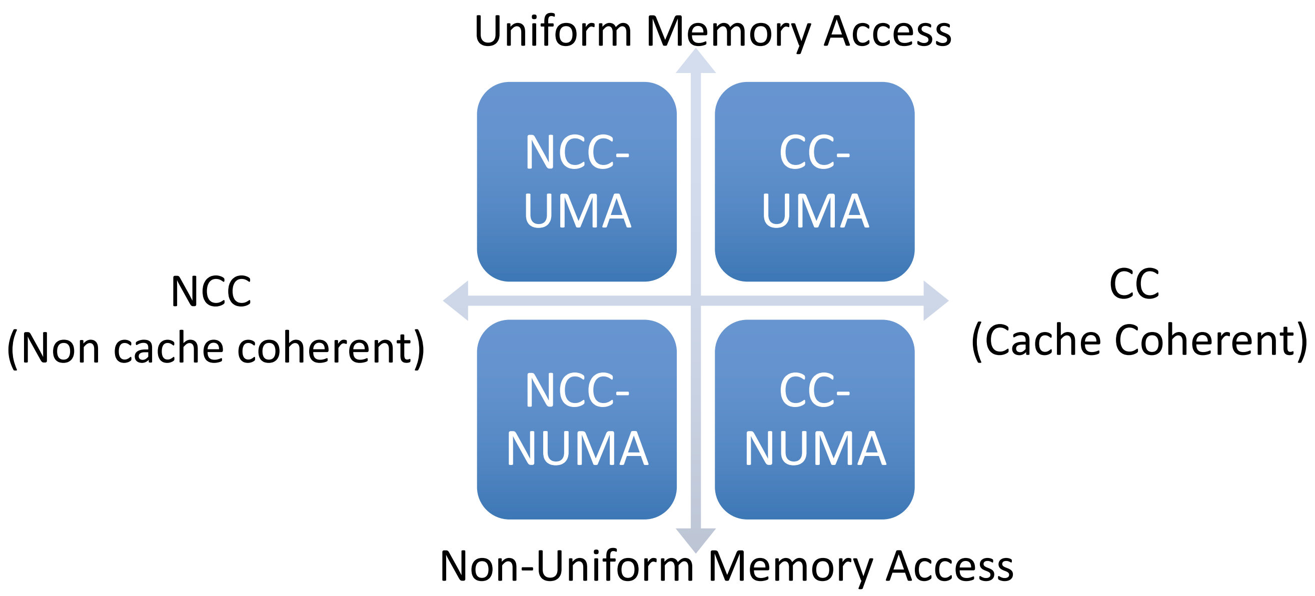


Shared Memory MIMD (contd.)

- Advantages:
  - No need to partition code or data
  - No need to physically move data among processors
    - communication is efficient
- Disadvantages:
  - Special synchronization constructs are required
  - Lack of scalability due to contention

Shared Memory MIMD Classification

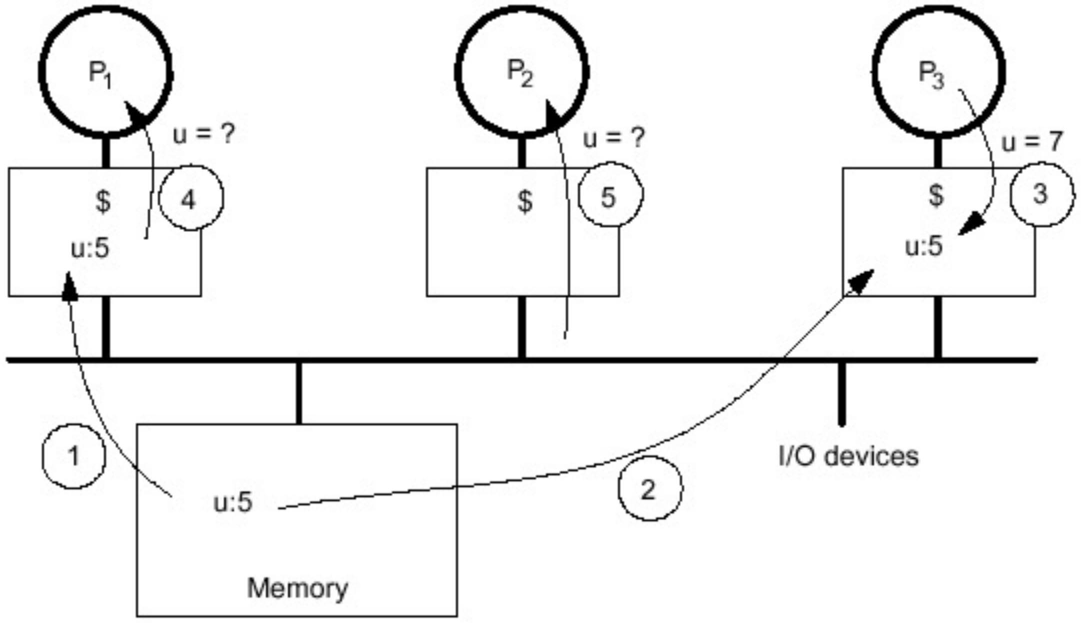
- Processor has cache (CC / NCC)
  - Same shared variable may exist in multiple caches
  - Hardware ensures correctness via cache coherence protocol
- Processor to memory delay (UMA / NUMA)
  - All accesses to memory encounter same delay or not [uniform versus non-uniform memory access]



Cache Coherence

- Caches reduce data access time and bandwidth requirement on the interconnect
- But private caches create problem
  - Copies of a variable in multiple caches
  - A write by one processor may not become visible to others
    - They may keep accessing stale data in their caches
  - Cache coherence problem

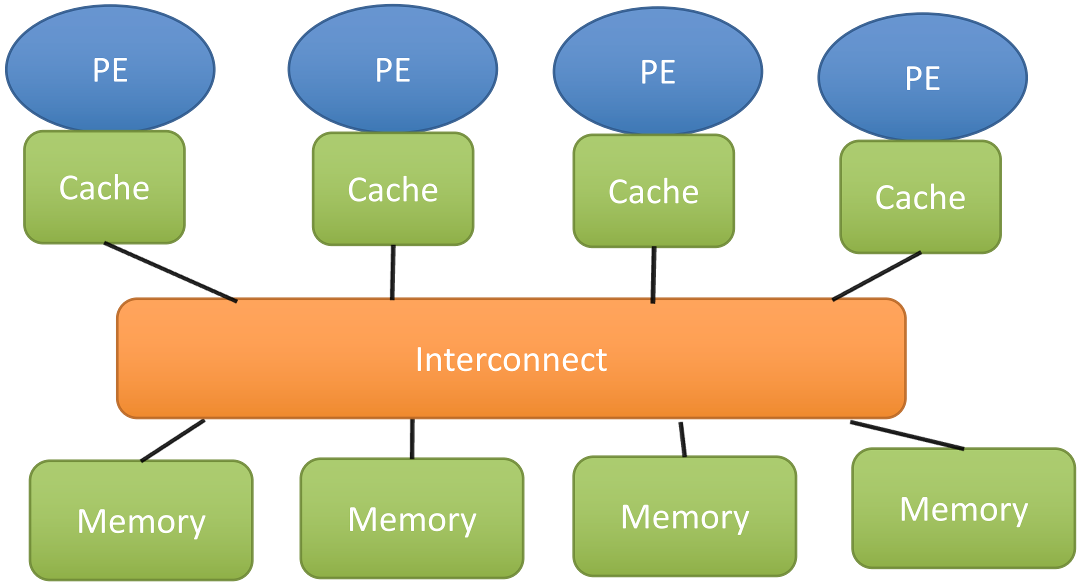
Cache Coherence Problem



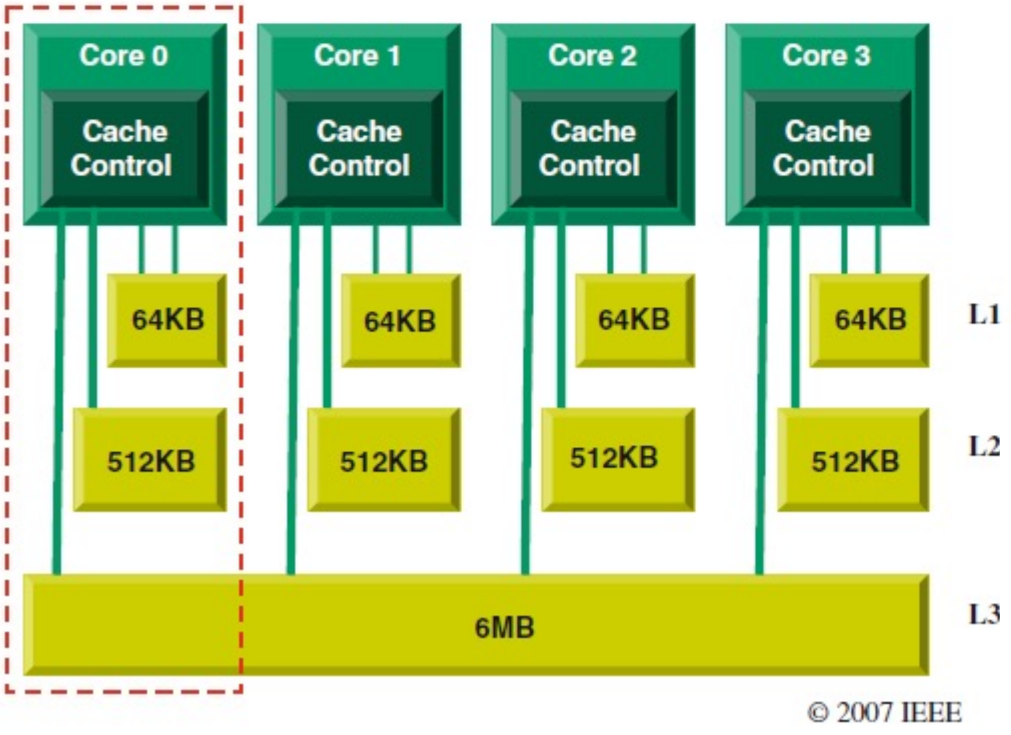
- Processors see different values of u after event 3
- With write-back caches, main memory will have the stale value till the entry is replaced

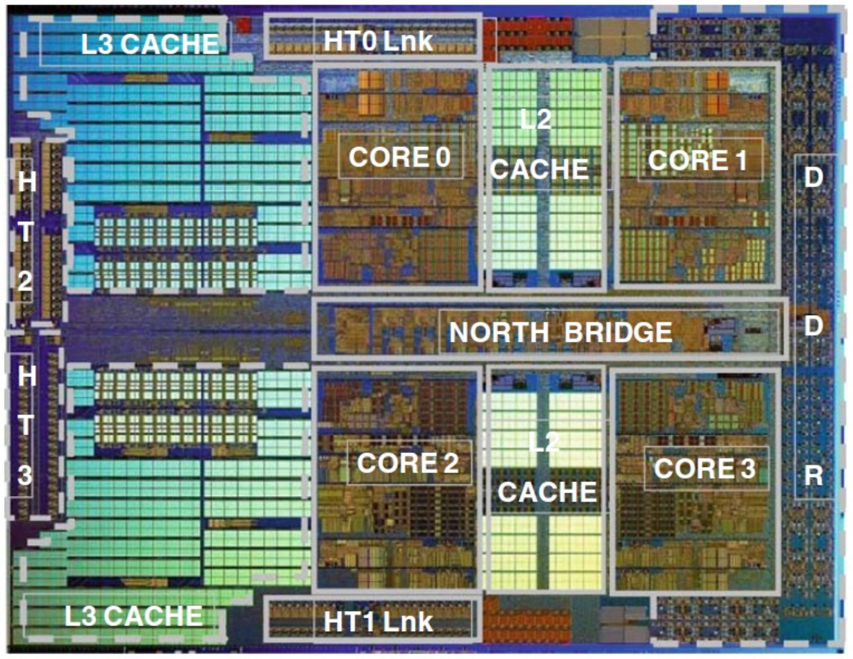
CC-UMA or SMP

- Cache-Coherent Uniform Memory Access
- Also known as Symmetric Multiprocessor (SMP)

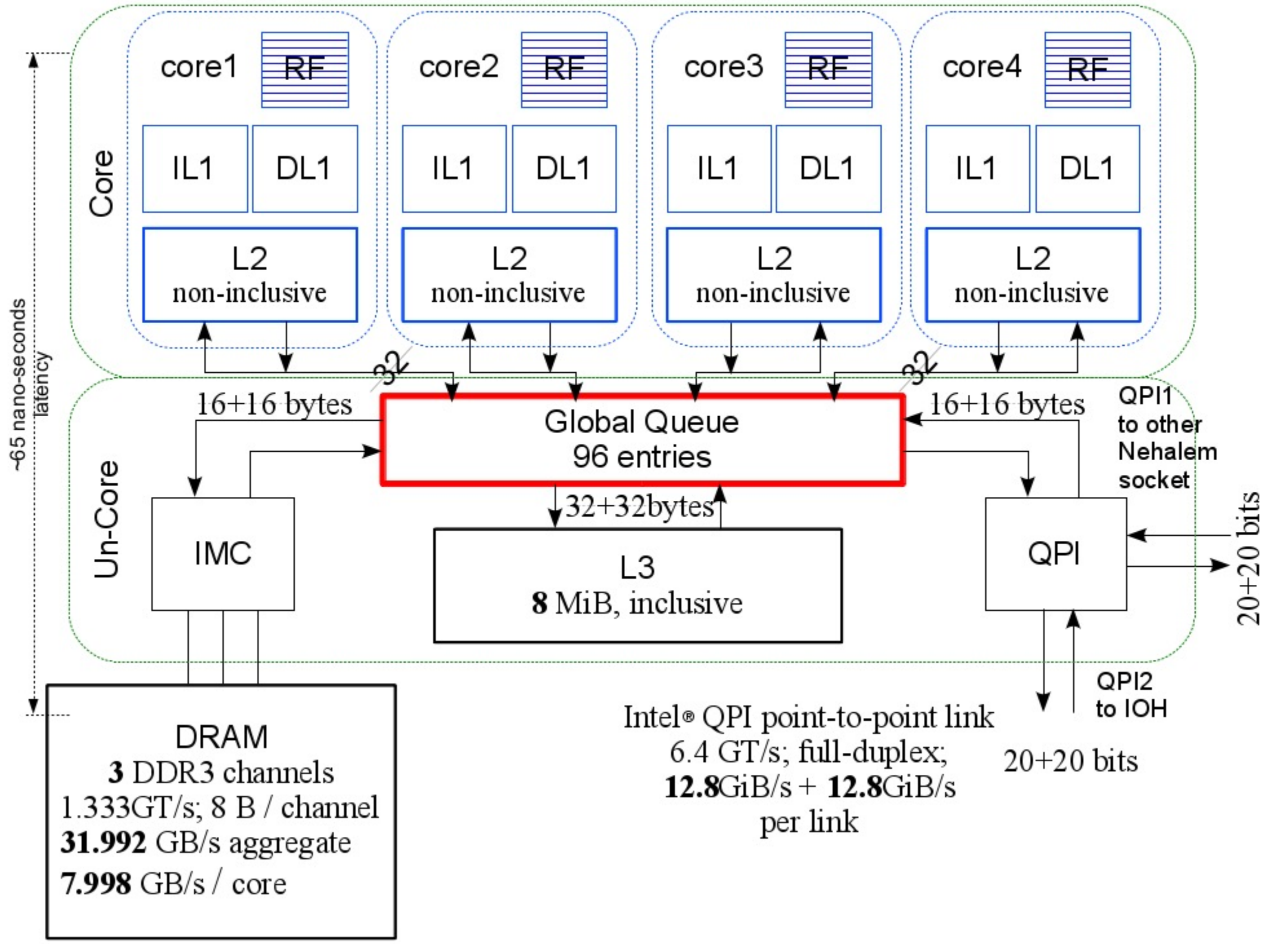


Contemporary CC-UMA: AMD Opteron



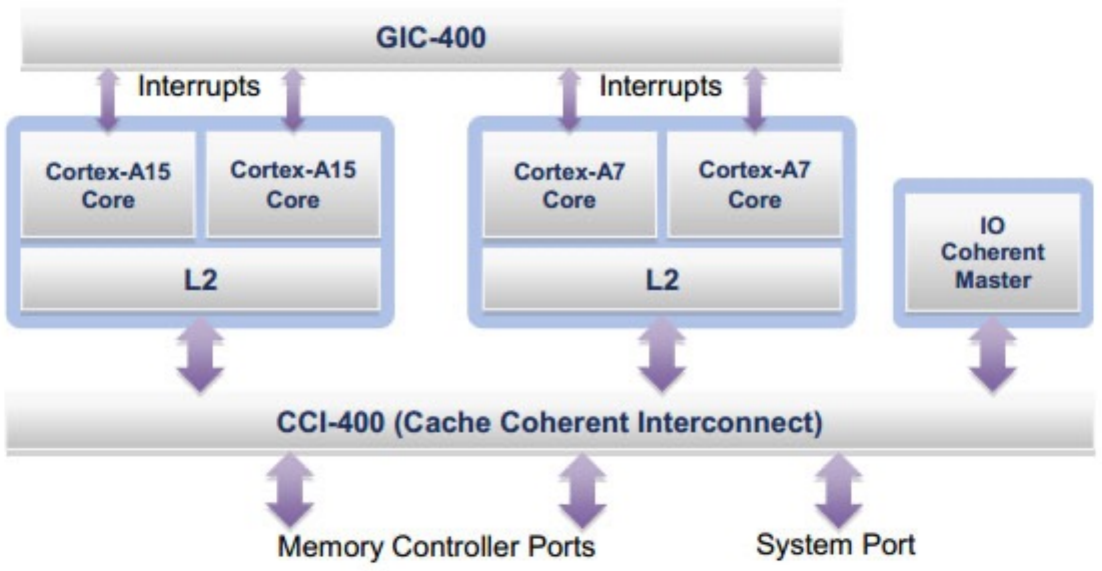


Contemporary CC-UMA: Intel Nehalem

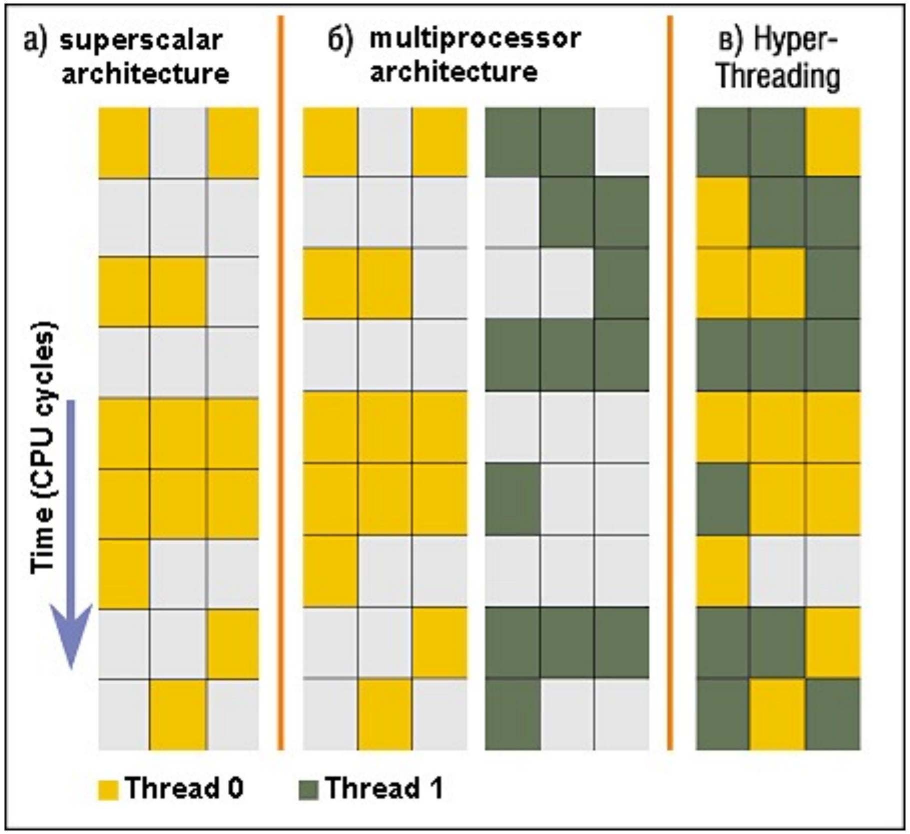


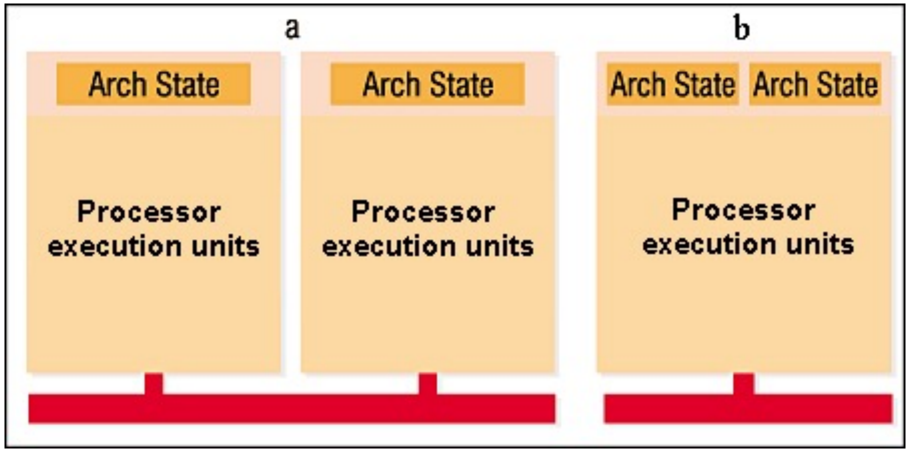
core1
core2
core3
core4
RF
IL1
DL1
L2 non-inclusive
Core
Un-Core
~65 nano-seconds latency
32
16+16 bytes
Global Queue 96 entries
32+32bytes
IMC
L3 8 MiB, inclusive
QPI
QPI1 to other Nehalem socket
20+20 bits
QPI2 to IOH
DRAM 3 DDR3 channels 1.333GT/s; 8 B / channel 31.992 GB/s aggregate 7.998 GB/s / core
Intel® QPI point-to-point link 6.4 GT/s; full-duplex; 12.8GiB/s + 12.8GiB/s per link

Mobile CC-UMA: ARM big.LITTLE



Simultaneous Multi-threading SMT (a.k.a. Hyper-threading)



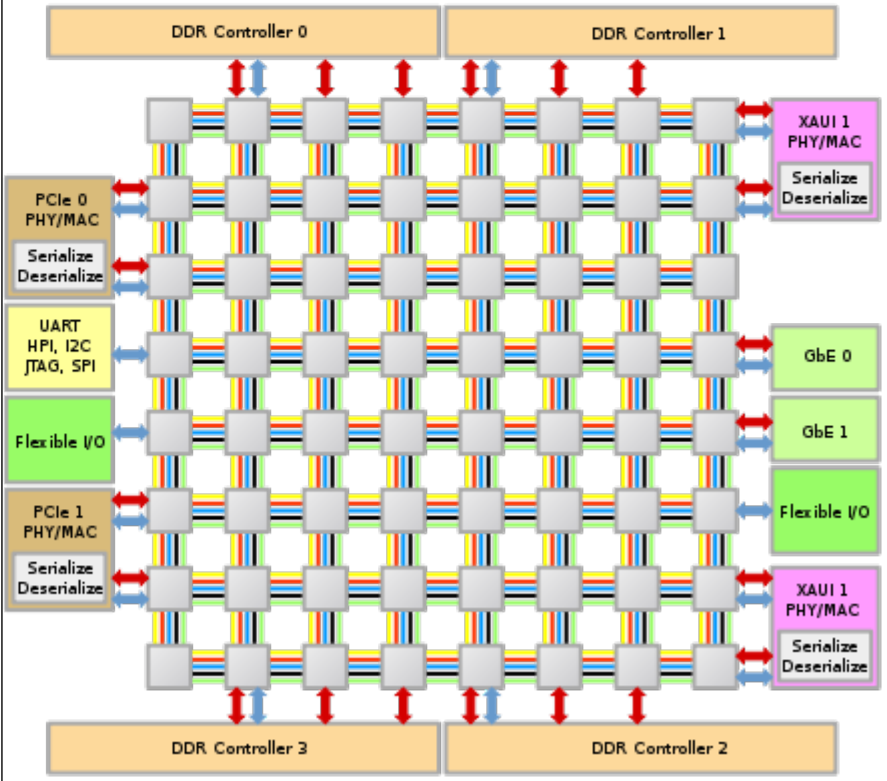


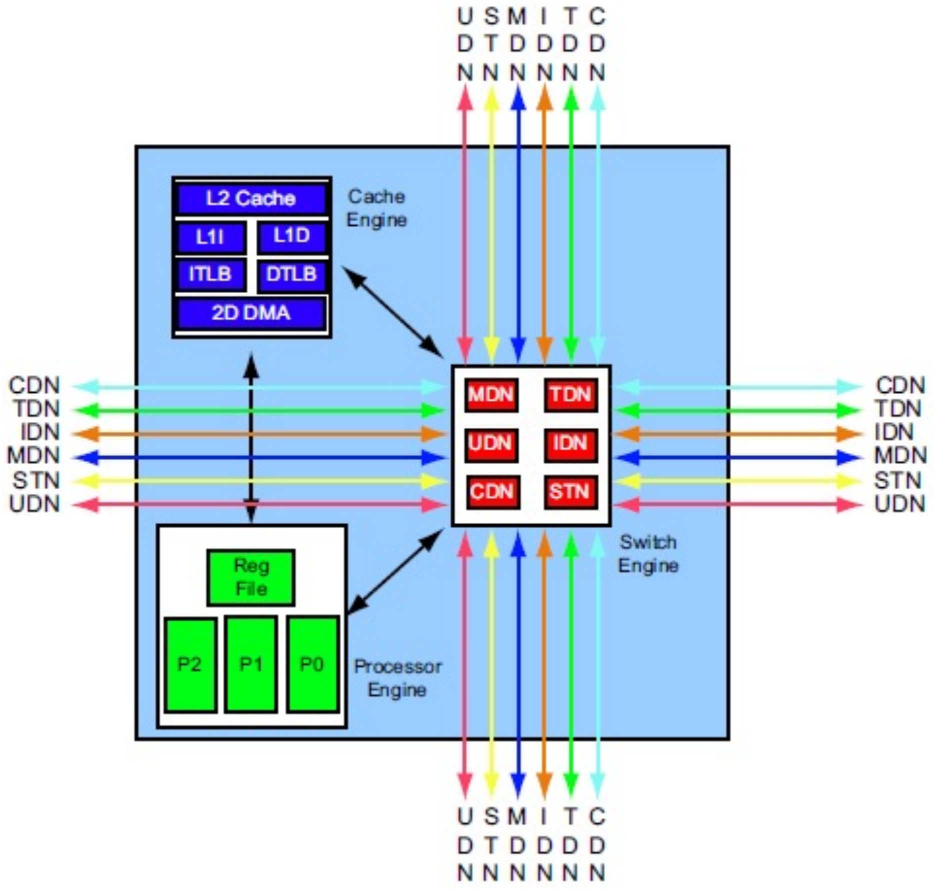
a
b
Arch State
Processor execution units
Arch State
Processor execution units
Arch State
Arch State
Processor execution units

Hyper-threading demo

https://www.youtube.com/watch?v=ptnotrfmA1E

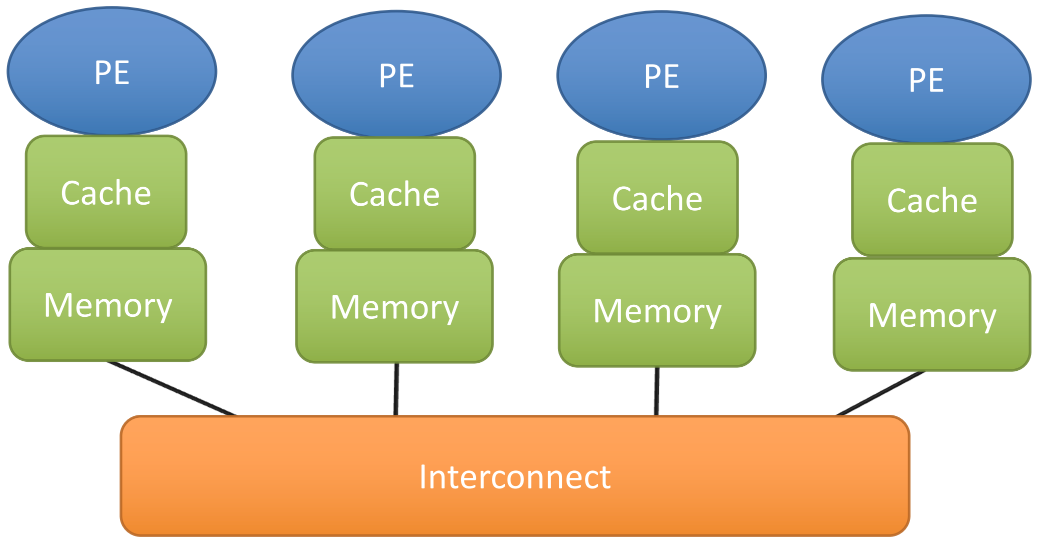
CC-UMA: Scaling to Many Core Tilera



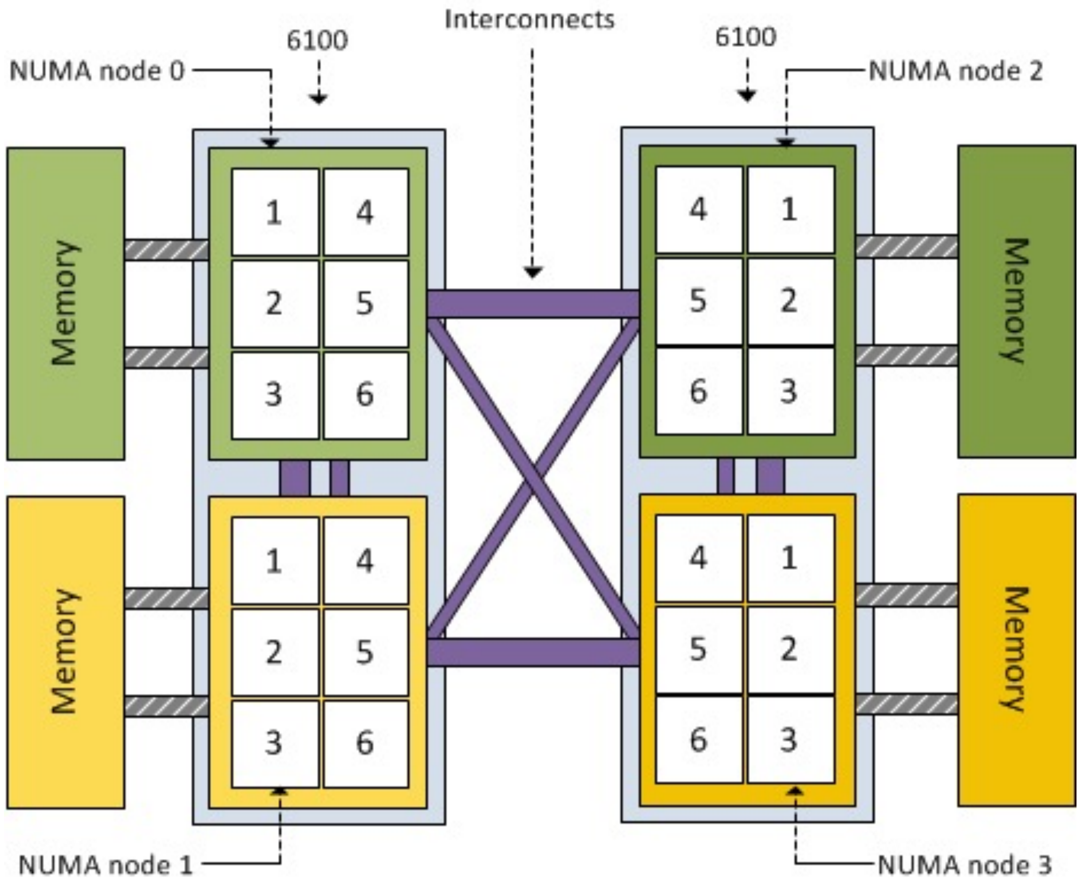


CC-NUMA

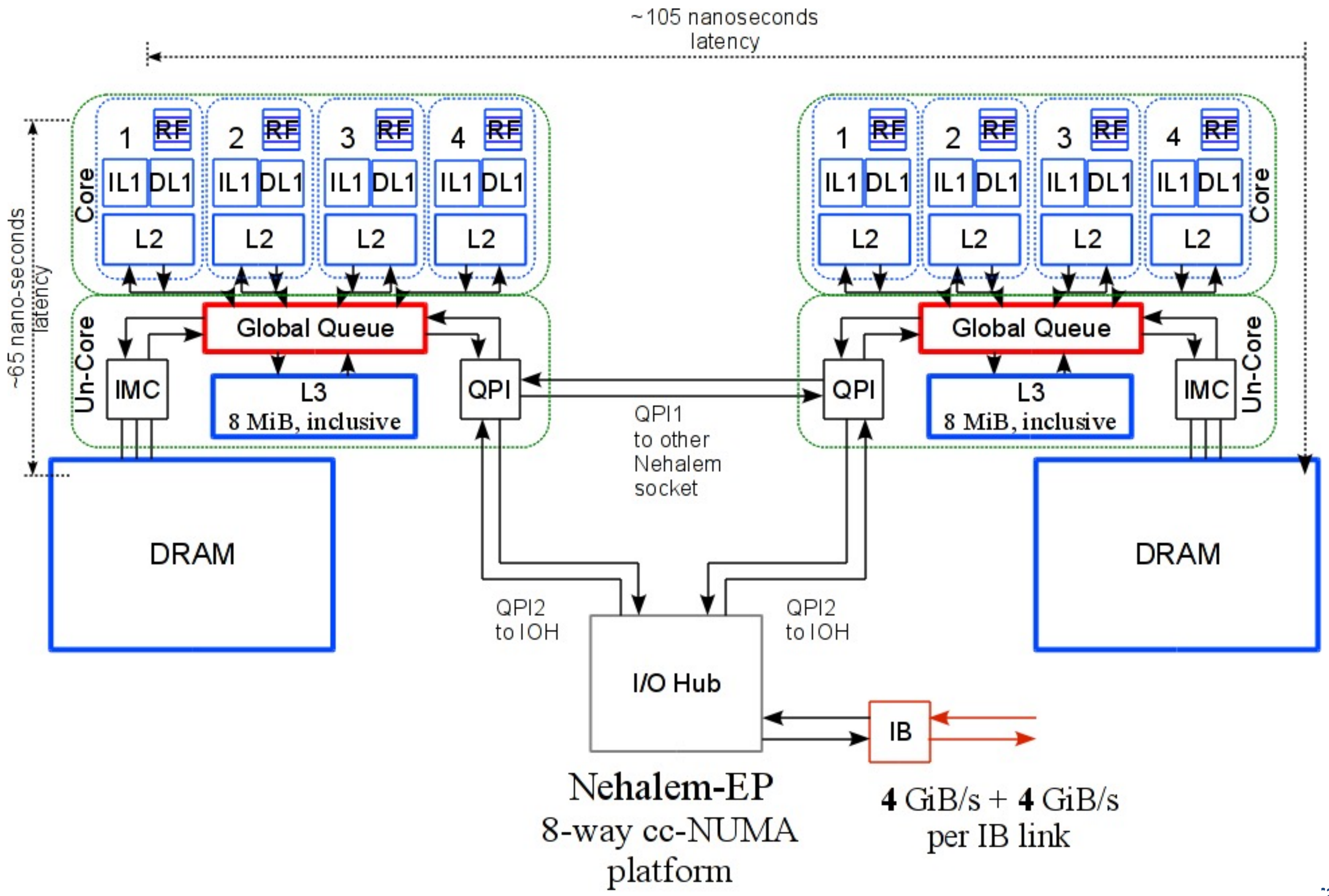
- Cache Coherent Non Uniform Memory Access
- Each node has cache memory to reduce contention



cc-NUMA: AMD 6100 Opteron package

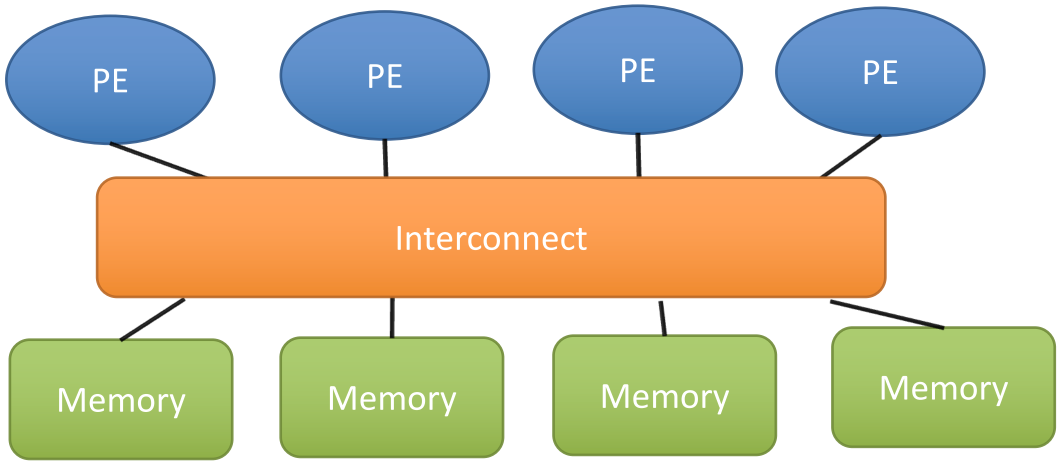


CC-NUMA: Nehalem Cluster



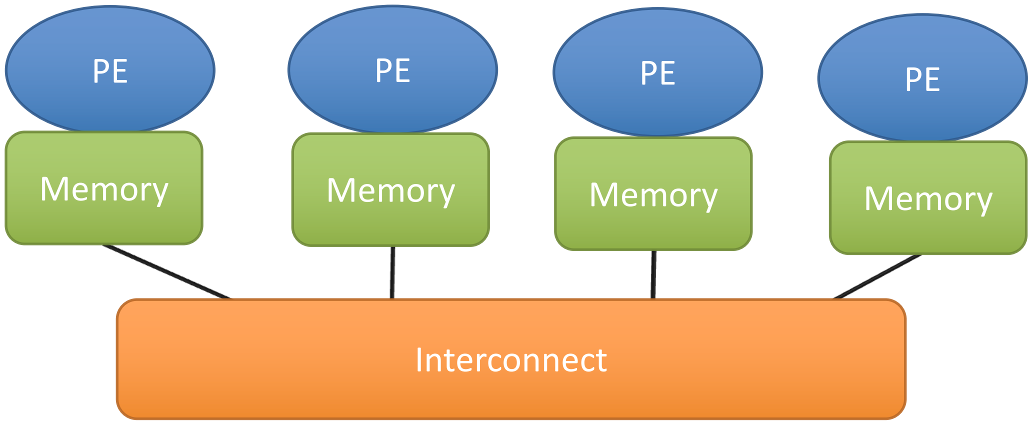
NCC-UMA

- Non cache-coherent uniform memory access
- Too much memory contention
- Does not make sense



NCC-NUMA

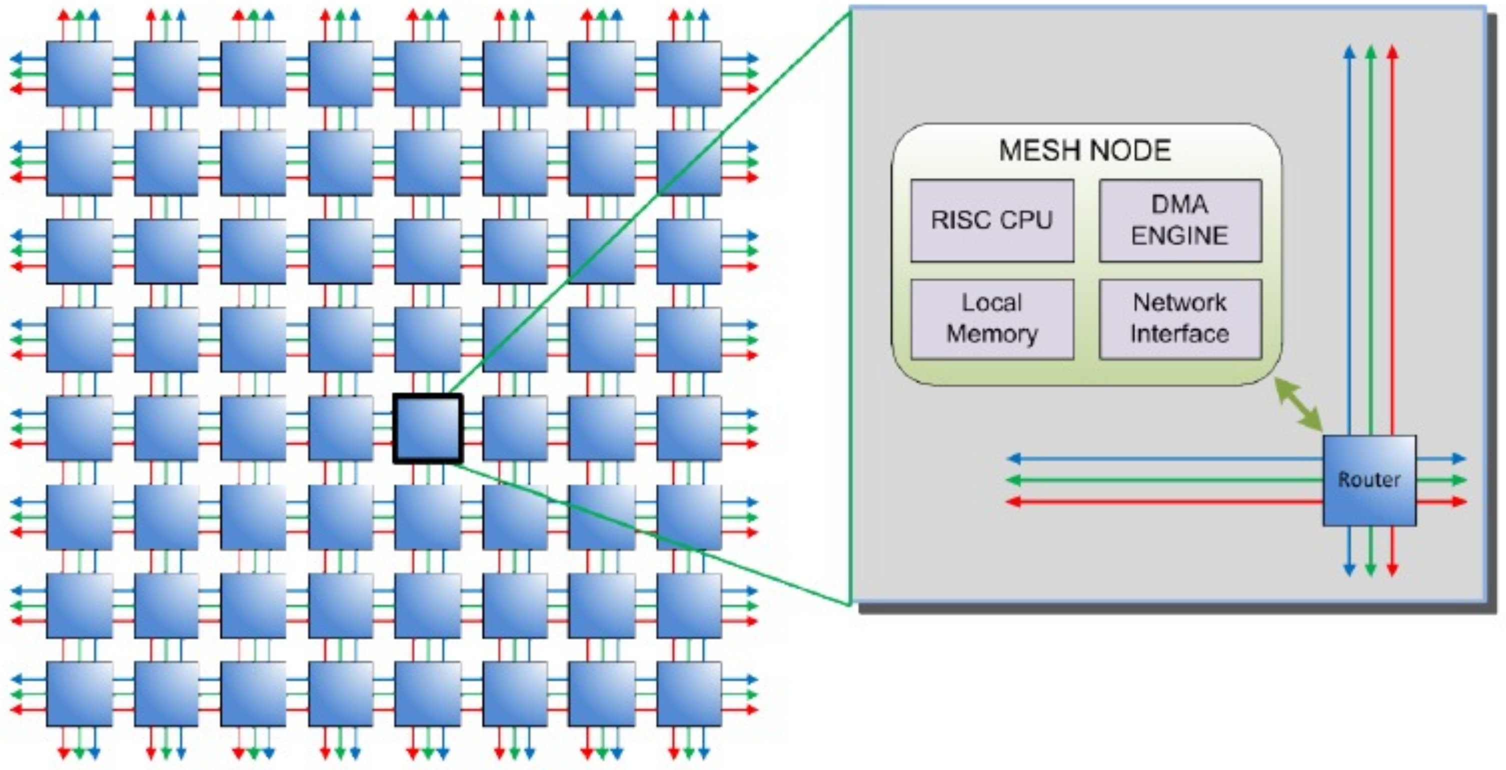
- Non Cache-Coherent Non-Uniform Memory Access
- Memory is logically shared, physically distributed
- Requires careful load balancing and data distribution just as distributed memory MIMD; programming is still based on shared memory



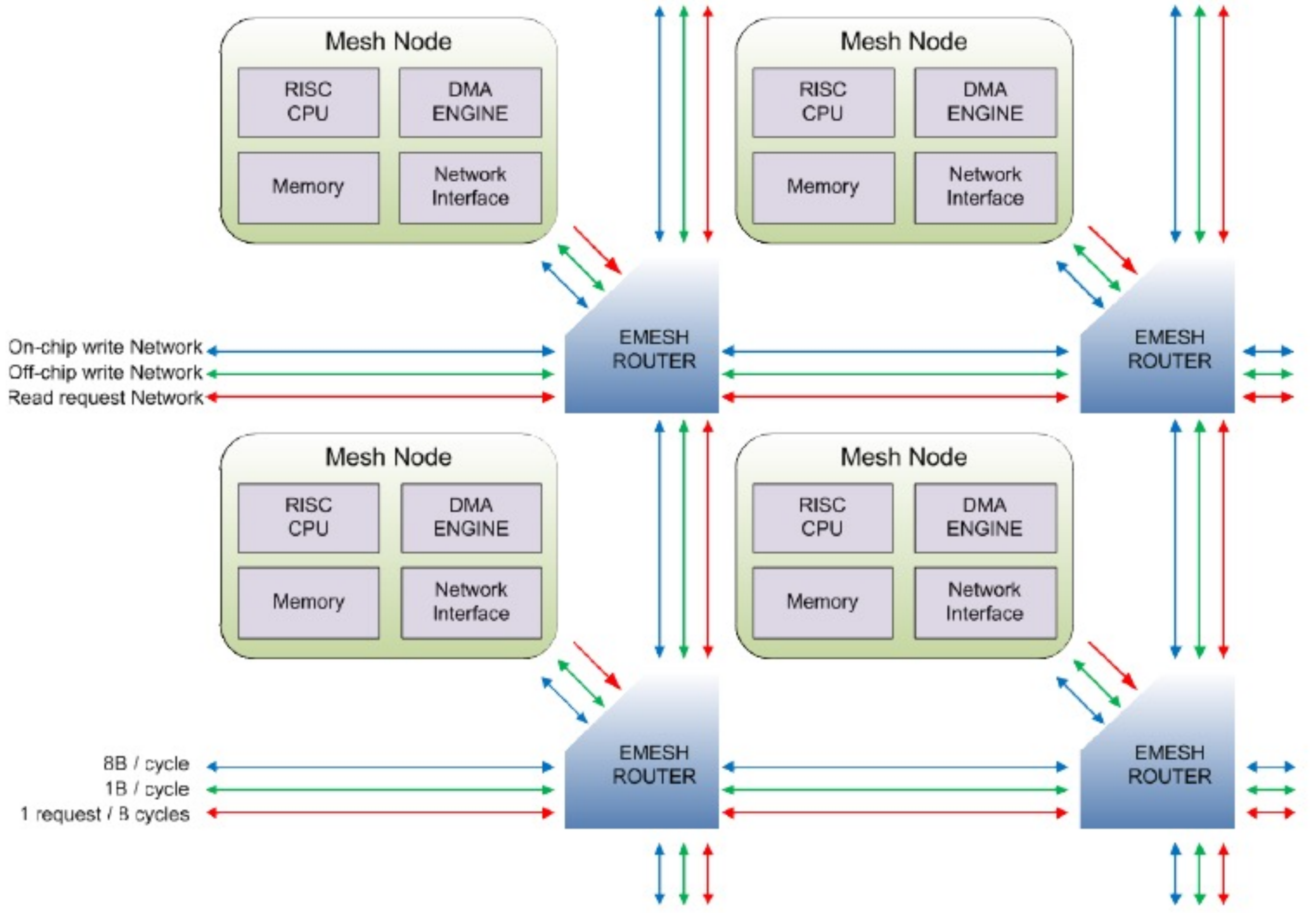
Distributed shared memory MIMD vs. Distributed memory MIMD

- Physical construction is the same
- In distributed shared memory MIMD (NUMA), local memories are components of global address space and any processor can access the local memory of any other processor directly
- In distributed memory MIMD, local memories have separate address spaces, and direct access to the local memory of a remote processor is prohibited

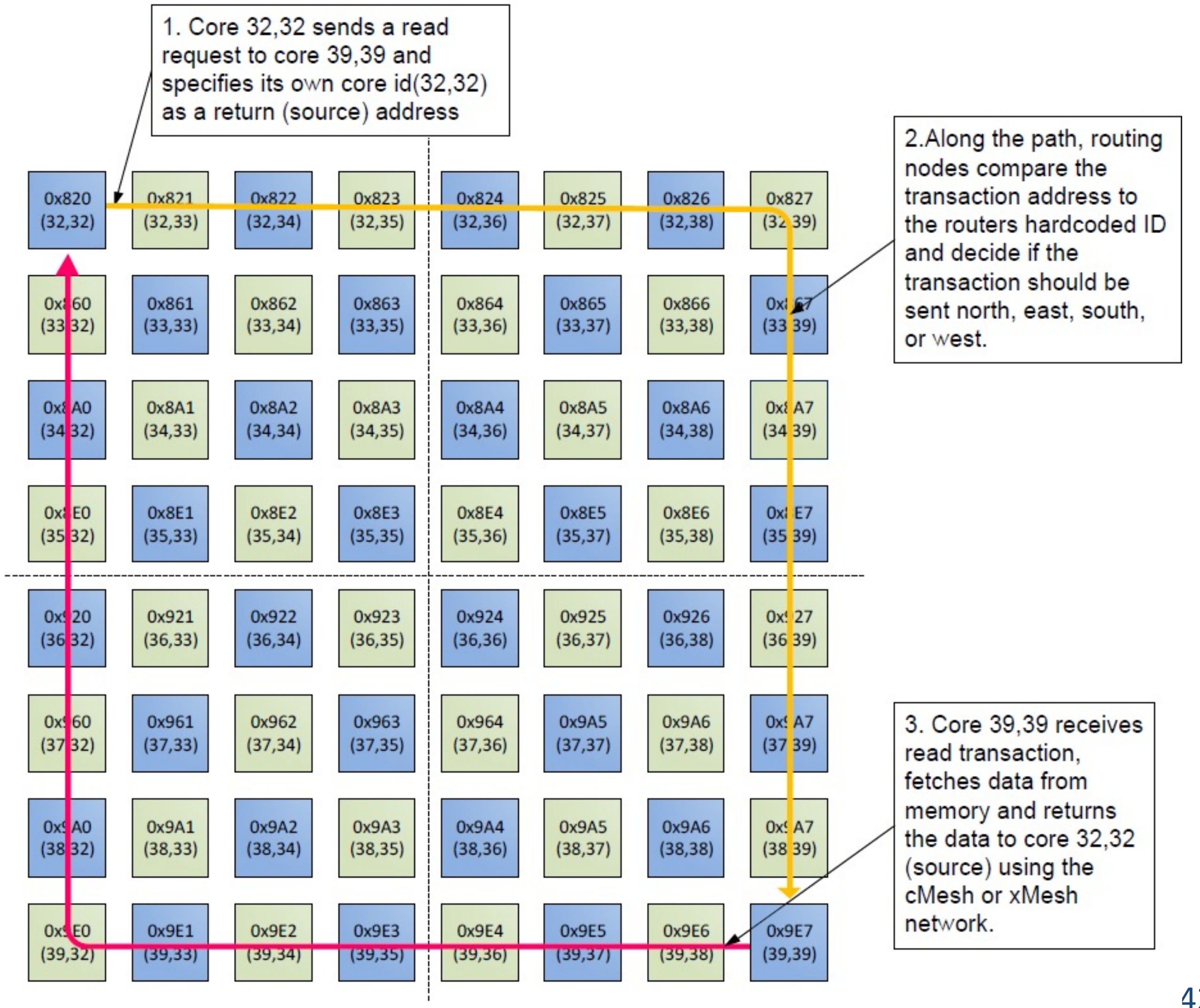
There's Something about Many: Parallela Epiphany Architecture



Network-on-chip



Routing


1. Core 32,32 sends a read request to core 39,39 and specifies its own core id(32,32) as a return (source) address

2.Along the path, routing nodes compare the transaction address to the routers hardcoded ID and decide if the transaction should be sent north, east, south, or west.

3. Core 39,39 receives read transaction, fetches data from memory and returns the data to core 32,32 (source) using the cMesh or xMesh network.

Making of Intel Core i7

http://www.youtube.com/watch?v=tKX8bdHWgu8