Parallel Architectures Overview

Krste Asanovic

krste@lcs.mit.edu

http://www.cag.lcs.mit.edu/6.893-f2000/

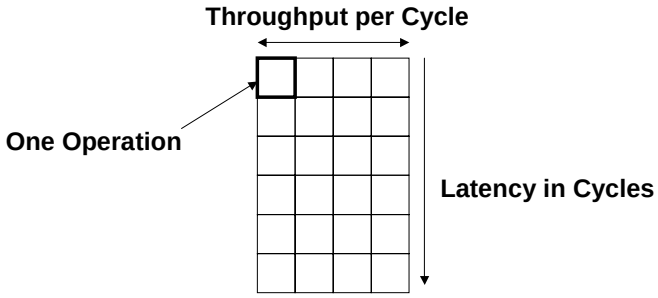
Parallelism is the "Ketchup* of Computer Architecture"

Apply parallelism to a hardware problem and it usually tastes better

- **Long latencies in off-chip memory and on-chip interconnect??** ***Interleave parallel operations to hide latencies***
- **High power consumption??** ***Use parallelism to increase throughput and use voltage scaling to reduce energy/operation***
- **Problem is finding and exploiting application parallelism**
- **Biggest problem is software**

[*substitute condiment of choice…]

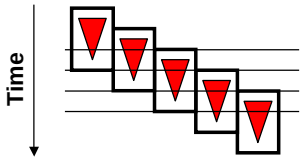
Little's Law



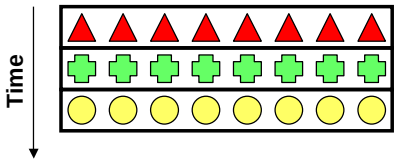
Parallelism = Throughput * Latency

- To maintain throughput T/cycle when each operation has latency L cycles, need T*L *independent* operations
- For fixed parallelism:
  - decreased latency allows increased throughput
  - decreased throughput allows increased latency tolerance

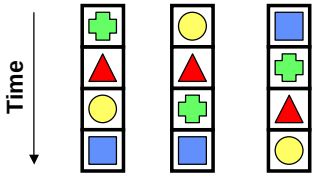
Types of Parallelism



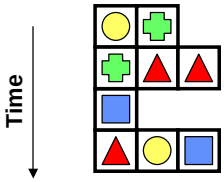
Pipelining



Data-Level Parallelism (DLP)



Thread-Level Parallelism (TLP)

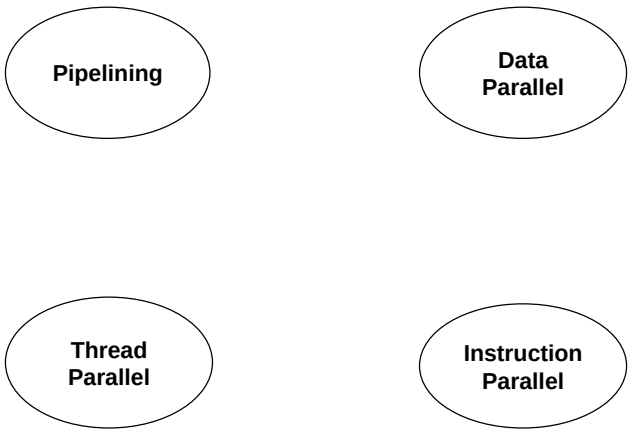


Instruction-Level Parallelism (ILP)

Dual Processor Pentium-III Desktop

- **Pipelined**
  - **minimum of 11 stages for any instruction**
- **Data Parallel Instructions**
  - **MMX (64-bit) and SSE (128-bit) extensions provide short vector support**
- **Instruction-Level Parallel Core**
  - **Can execute up to 3 x86 instructions per cycle**
- **Thread-Level Parallelism at System Level**
  - **Bus architecture supports shared memory multiprocessing**

Translating Parallelism Types



Speculation and Parallelism

- Speculation can increase effective parallelism by avoiding true dependencies
  - branch prediction for control flow speculation
  - address prediction for memory access reordering
  - value prediction to avoid operation latency
- Requires mechanism to recover on mispredict

Issues in Parallel Machine Design

- Communication
  - how do parallel operations communicate data results?
- Synchronization
  - how are parallel operations coordinated?
- Resource Management
  - how are a large number of parallel tasks scheduled onto finite hardware?
- Scalability
  - how large a machine can be built?

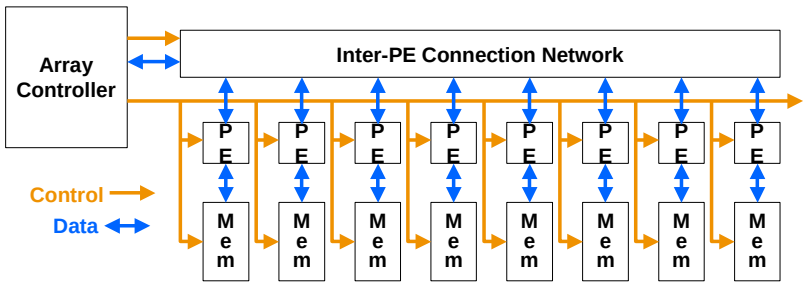
Flynn's Classification (1966)

Broad classification of parallel computing systems based on number of instruction and data streams

- **SISD: Single Instruction, Single Data**
  - conventional uniprocessor
- **SIMD: Single Instruction, Multiple Data**
  - one instruction stream, multiple data paths
  - distributed memory SIMD (MPP, DAP, CM-1&2, Maspar)
  - shared memory SIMD (STARAN, vector computers)
- **MIMD: Multiple Instruction, Multiple Data**
  - message passing machines (Transputers, nCube, CM-5)
  - non-cache-coherent shared memory machines (BBN Butterfly, T3D)
  - cache-coherent shared memory machines (Sequent, Sun Starfire, SGI Origin)
- **MISD: Multiple Instruction, Single Data**
  - no commercial examples

SIMD Architecture

- **Central controller broadcasts instructions to multiple processing elements (PEs)**

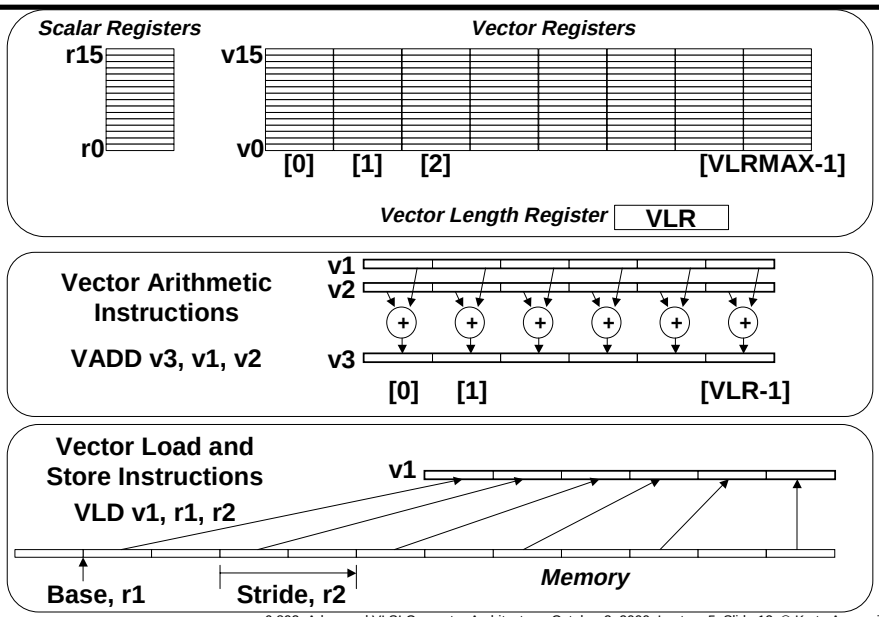


- Only requires one controller for whole array
- Only requires storage for one copy of program
- All computations fully synchronized

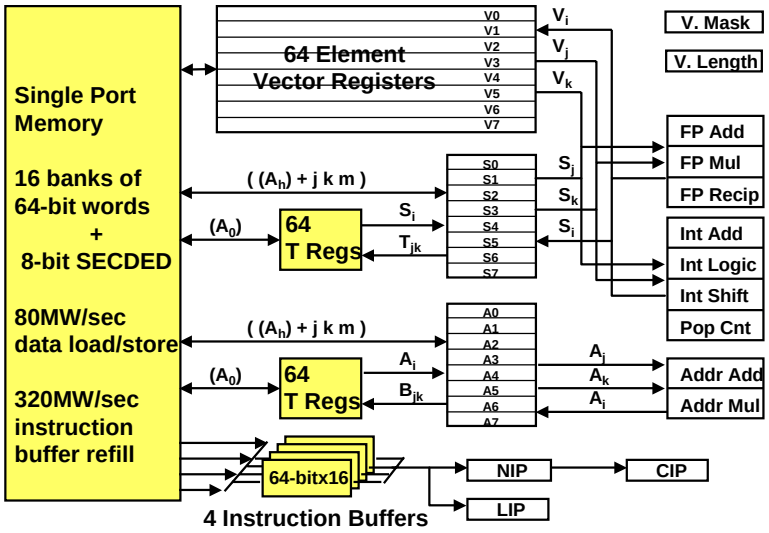
SIMD Machines

- Illiac IV (1972)
  - 64 64-bit PEs, 16KB/PE, 2D network
- Goodyear STARAN (1972)
  - 256 bit-serial associative PEs, 32B/PE, multistage network
- ICL DAP (Distributed Array Processor) (1980)
  - 4K bit-serial PEs, 512B/PE, 2D network
- Goodyear MPP (Massively Parallel Processor) (1982)
  - 16K bit-serial PEs, 128B/PE, 2D network
- Thinking Machines Connection Machine CM-1 (1985)
  - 64K bit-serial PEs, 512B/PE, 2D + hypercube router
  - CM-2: 2048B/PE, plus 2,048 32-bit floating-point units
- Maspar MP-1 (1989)
  - 16K 4-bit processors, 16-64KB/PE, 2D + Xnet router
  - MP-2: 16K 32-bit processors, 64KB/PE

Vector Register Machine

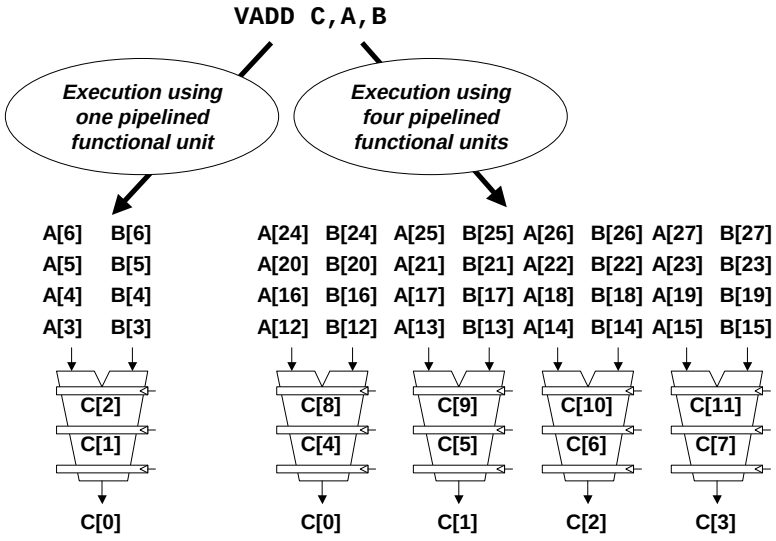


Cray-1 (1976)

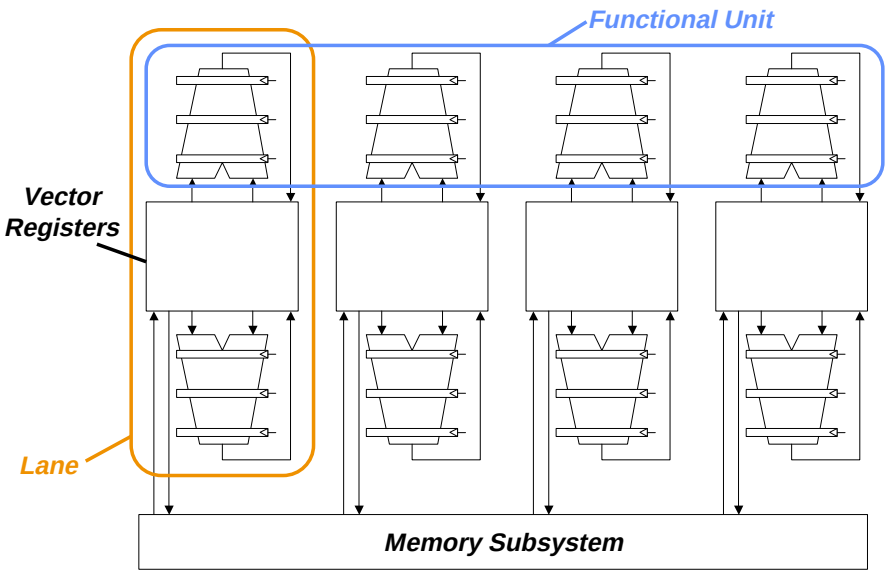


memory bank cycle 50 ns *processor cycle* 12.5 ns (80MHz)

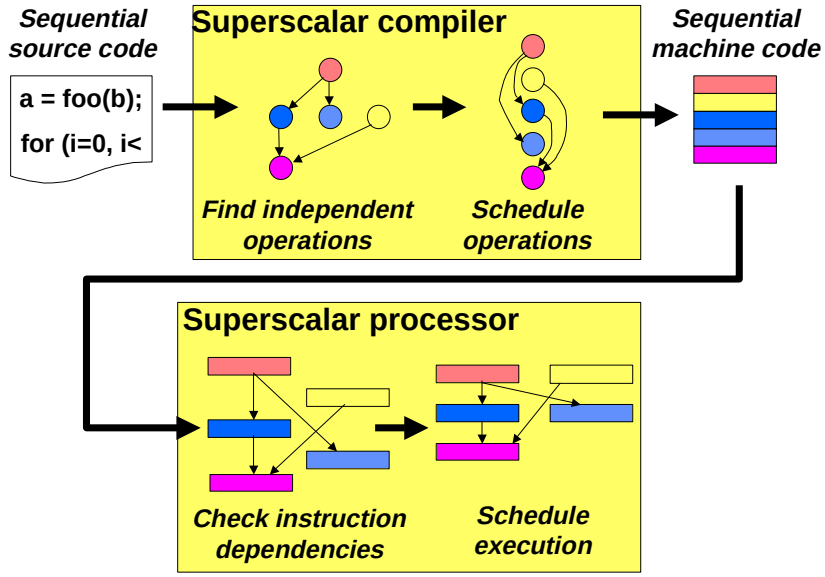
Vector Instruction Execution



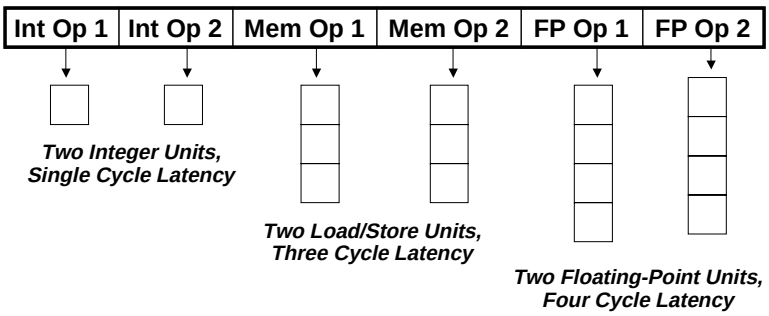
Vector Unit Structure



Sequential ISA Bottleneck

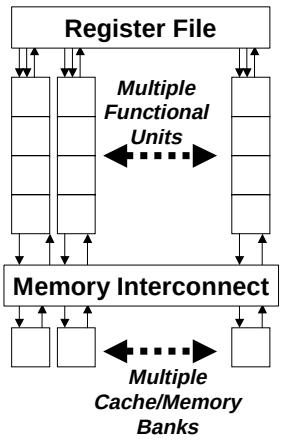


VLIW: Very Long Instruction Word



- Compiler schedules parallel execution
- Multiple parallel operations packed into one long instruction word
- Compiler must avoid data hazards (no interlocks)

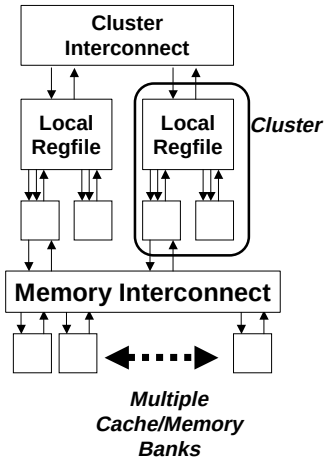
ILP Datapath Hardware Scaling



- Replicating functional units and cache/memory banks is straightforward and scales linearly
- Register file ports and bypass logic for N functional units scale quadratically (N*N)
- Memory interconnection among N functional units and memory banks also scales quadratically
- *(For large N, could try O(N logN) interconnect schemes)*
- Technology scaling: Wires are getting even slower relative to gate delays
- Complex interconnect adds latency as well as area

  => Need greater parallelism to hide latencies

Clustered VLIW



- Divide machine into clusters of local register files and local functional units
- Lower bandwidth/higher latency interconnect between clusters
- Software responsible for mapping computations to minimize communication overhead

MIMD Machines

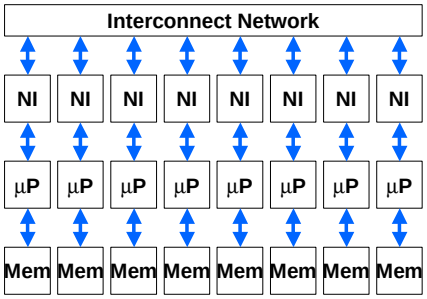
- **Message passing**
  - Thinking Machines CM-5
  - Intel Paragon
  - Meiko CS-2
  - many cluster systems (e.g., IBM SP-2, Linux Beowulfs)
- **Shared memory**
  - no hardware cache coherence
    - IBM RP3
    - BBN Butterfly
    - Cray T3D/T3E
    - Parallel vector supercomputers (Cray T90, NEC SX-5)
  - hardware cache coherence
    - many small-scale SMPs (e.g. Quad Pentium Xeon systems)
    - large scale bus/crossbar-based SMPs (Sun Starfire)
    - large scale directory-based SMPs (SGI Origin)

Message Passing MPPs

(Massively Parallel Processors)

- Initial Research Projects
  - Caltech Cosmic Cube (early 1980s) using custom Mosaic processors
- Commercial Microprocessors including MPP Support
  - Transputer (1985)
  - nCube-1(1986) /nCube-2 (1990)
- Standard Microprocessors + Network Interfaces
  - Intel Paragon (i860)
  - TMC CM-5 (SPARC)
  - Meiko CS-2 (SPARC)
  - IBM SP-2 (RS/6000)
- MPP Vector Supers
  - Fujitsu VPP series

Designs scale to 100s or 1000s of nodes



Message Passing MPP Problems

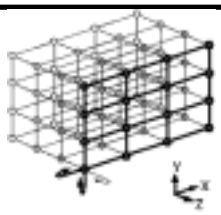
- All data layout must be handled by software
  - cannot retrieve remote data except with message request/reply
- Message passing has high software overhead
  - early machines had to invoke OS on each message (100µs-1ms/message)
  - even user level access to network interface has dozens of cycles overhead (NI might be on I/O bus)
  - sending messages can be cheap (just like stores)
  - receiving messages is expensive, need to poll or interrupt

Shared Memory Multiprocessors

- Will work with any data placement (but might be slow)
  - can choose to optimize only critical portions of code
- Load and store instructions used to communicate data between processes
  - no OS involvement
  - low software overhead
- Usually some special synchronization primitives
  - fetch&op
  - load linked/store conditional
- In large scale systems, the logically shared memory is implemented as physically distributed memory modules
- Two main categories
  - non cache coherent
  - hardware cache coherent

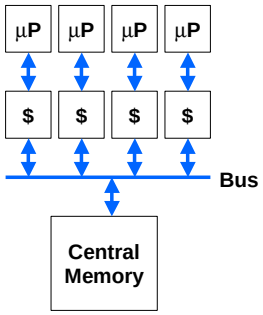
Cray T3E

• Up to 2048 600MHz Alpha 21164 processors connected in 3D torus



- Each node has 256MB-2GB local DRAM memory
- Load and stores access global memory over network
- Only local memory cached by on-chip caches
- Alpha microprocessor surrounded by custom “shell” circuitry to make it into effective MPP node. Shell provides:
  - multiple stream buffers instead of board-level (L3) cache
  - external copy of on-chip cache tags to check against remote writes to local memory, generates on-chip invalidates on match
  - 512 external E registers (asynchronous vector load/store engine)
  - address management to allow all of external physical memory to be addressed
  - atomic memory operations (fetch&op)
  - support for hardware barriers/eureka to synchronize parallel tasks

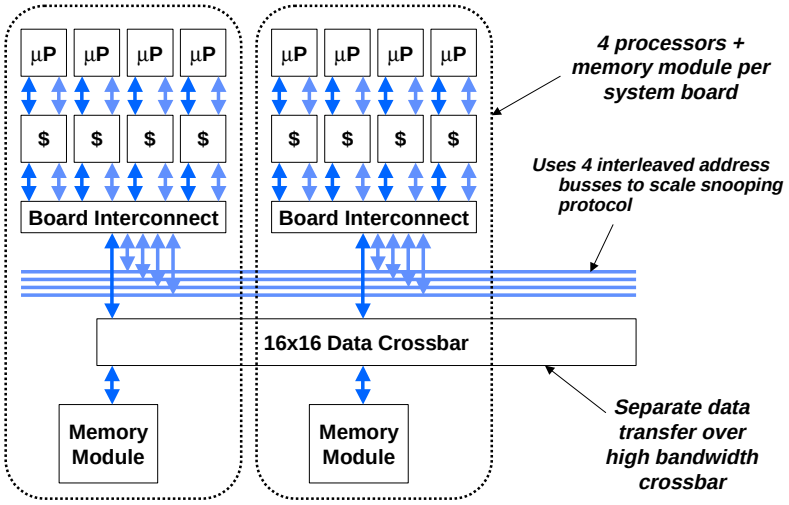
Bus-Based Cache-Coherent SMPs



- Small scale (<= 4 processors) bus-based SMPs by far the most common parallel processing platform today
- Bus provides broadcast and serialization point for simple snooping cache coherence protocol
- Modern microprocessors integrate support for this protocol

Sun Starfire (UE10000)

• Up to 64-way SMP using bus-based snooping protocol

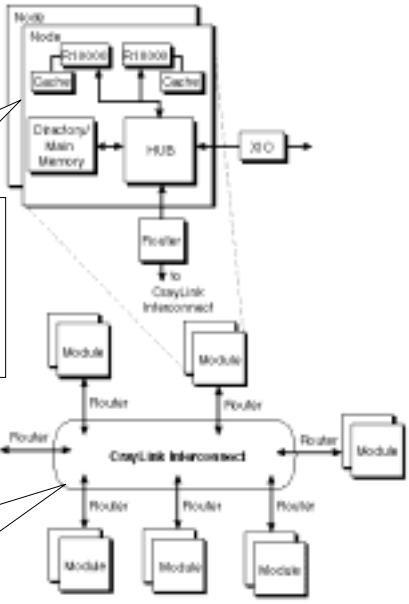


SGI Origin 2000

- **Large scale distributed directory SMP**
- **Scales from 2 processor workstation to 512 processor supercomputer**

Node contains:

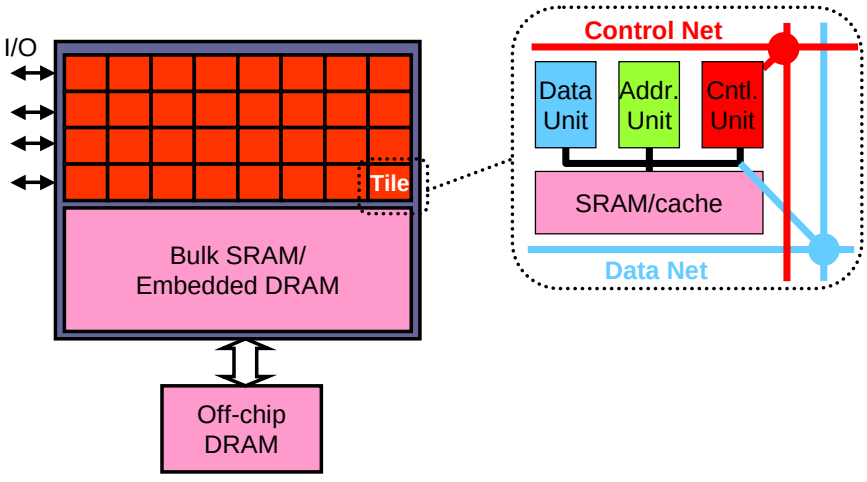
- ***Two MIPS R10000 processors plus caches***
- ***Memory module including directory***
- ***Connection to global network***
- ***Connection to I/O***



Scalable hypercube switching network supports up to 64 two-processor nodes (128 processors total)

(Some installations up to 512 processors)

Convergence in Parallel VLSI Architectures?



Portable Parallel Programming?

- **Most large scale commercial installations emphasize throughput**
  - **database servers, web servers, file servers**
  - **independent transactions**
- **Wide variety of parallel systems**
  - **message passing**
  - **shared memory**
  - **shared memory within node, message passing between nodes**

⇒ ***Little commercial software support for portable parallel programming***

Message Passing Interface (MPI) standard widely used for portability

- **lowest common denominator**
- **“assembly” language level of parallel programming**