ARCH I

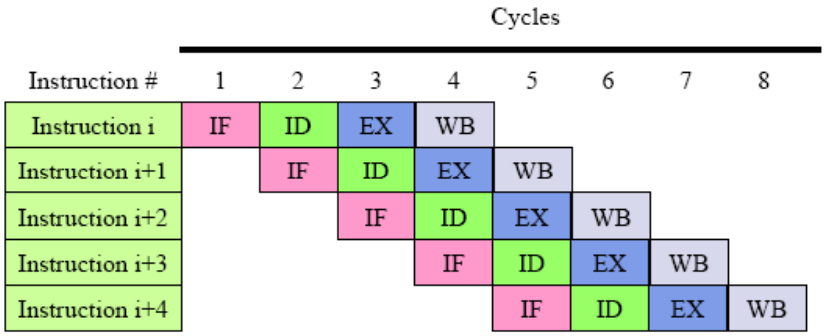
Overview of parallel architectures

Implicit Parallelism: Superscalar Processors

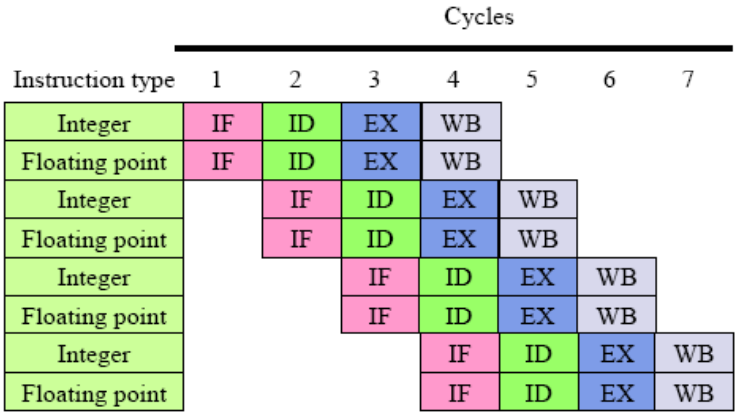
- Issue varying numbers of instructions per clock
  - statically scheduled
    - using compiler techniques
    - in-order execution
  - dynamically scheduled
    - Extracting ILP by examining 100's of instructions
    - Scheduling them in parallel as operands become available
    - Rename registers to eliminate anti dependences
    - out-of-order execution
    - Speculative execution

Pipelining Execution

IF: Instruction fetch
ID : Instruction decode
EX : Execution
WB : Write back



Super-Scalar Execution



2-issue super-scalar machine

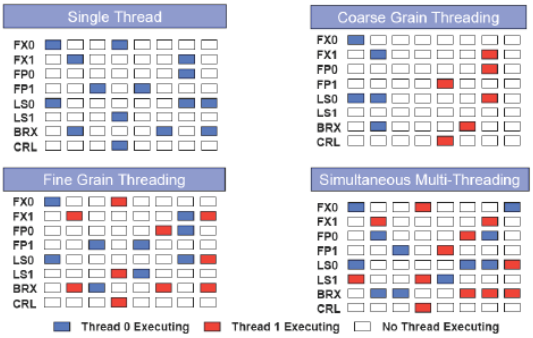
Speculation in Rampant in Modern Superscalars

- Different predictors
  - Branch Prediction
  - Value Prediction
  - Prefetching (memory access pattern prediction)
- Inefficient
  - Predictions can go wrong
  - Has to flush out wrongly predicted data
  - While not impacting performance, it consumes power
- Consequences
  - Low utilization
  - Long latencies

Multithreaded Architectures

- Designed to tolerate memory access latencies when a thread stalls for long-latency operations
- Multithreaded architectures allows for multiple threads to share the functional units of a single processor
  - Improves utilization of functional units w.r.t. superscalar machines

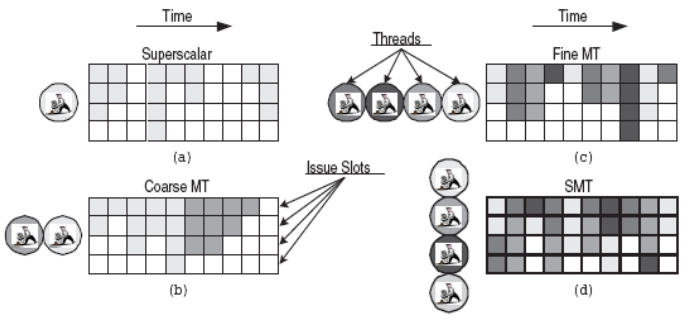
Threading Alternatives



- CGT: switches threads only with costly stalls such as L2 cache misses, problems with short stalls because of the start-up cost of the pipe
- FGT: switches at each clock cycle, hiding latency of short stalls, but slows down instructions ready to be executed
- SMT: multiple instructions from multiple threads in a single clock cycle (TLP+ILP)

SMT

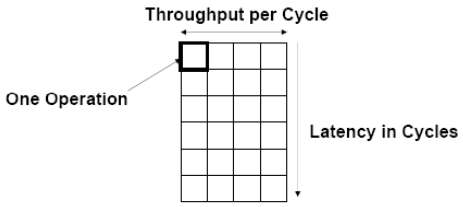
- Multiple instructions from independent threads can be issued
- Basic superscalar + resource replication to keep independent state (such as PCs)
  - Register files, instruction queues, branch predictors are shared



Explicit Parallel Processors

- Parallelism is exposed to software
  - Compiler or Programmer
- Many different forms
  - Loosely coupled Multiprocessors to tightly coupled VLIW
- We focus on MIMD architectures
  - TLP processors are MIMD
  - Shared network, shared memory
  - Chip multiprocessors (CMP)

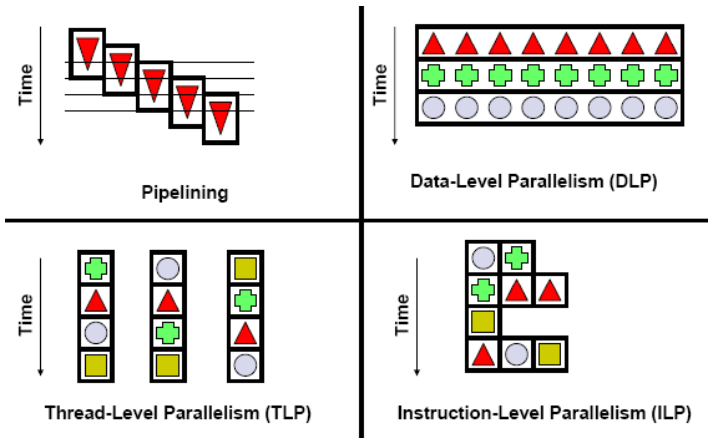
Little's Law



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 Paralellism

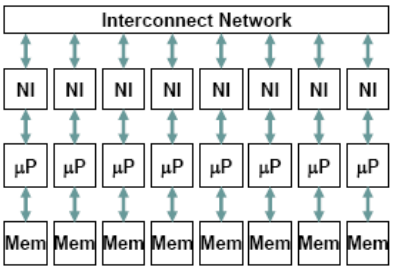


Issues in Parallel Machine Design

- 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?

Shared Network Processors (Massively Parallel Processors)

- Exploit message passing between cores
- 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 + Netw
  - Intel Paragon (i860)
  - TMC CM-5 (SPARC)
  - Meiko CS-2 (SPARC)
  - IBM SP-2 (RS/6000)
- MPP Vector Supers
  - Fujitsu VPP series
- 100s to 1000s nodes



Message Passing 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

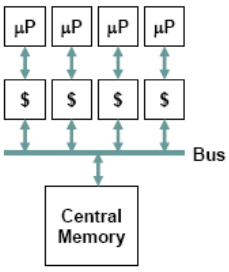
Cache Coherency

- 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)

HW Cache Coherency

- Bus-based Snooping Solution
  - Send all requests for data to all processors
  - Processors snoop to see if they have a copy and respond accordingly
  - Requires broadcast, since caching information is at processors
  - Works well with bus (natural broadcast medium)
  - Dominates for small scale machines (most of the market)
- Directory-Based Schemes
  - Keep track of what is being shared in 1 centralized place (logically)
  - Distributed memory => distributed directory for scalability (avoids bottlenecks)
  - Send point-to-point requests to processors via network
  - Scales better than Snooping
  - Actually existed BEFORE Snooping-based schemes

Bus-Based Cache-Coherent SMPs

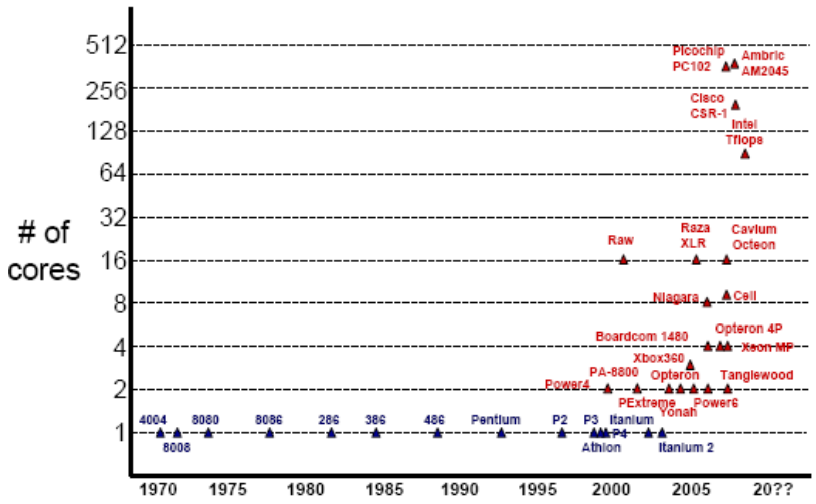


- Small scale (<= 4 processors) bus-based SMPs by far the most common parallel processing platform today
- Bus provides broadcast and serialization point for simple snooping cache coherence protocol
- Modern microprocessors integrate support for this protocol

CMP

- Full set of architectural resources on the same die
- It exploits TLP by executing different threads in parallel on different processors
- It consists of single-thread processor cores relatively simpler than general purpose processors

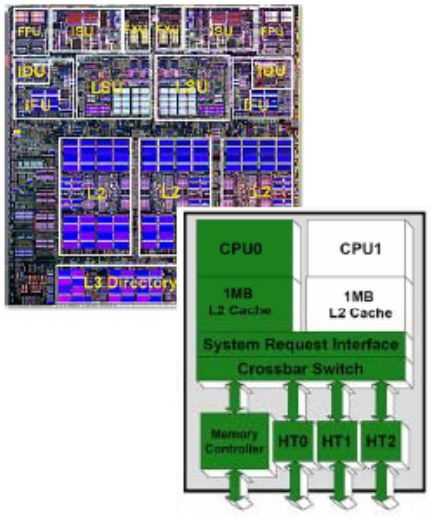
Multicores



Multicores

- Shared Memory
  - Intel Yonah, AMD Opteron
  - IBM Power 5 & 6
  - Sun Niagara
- Shared Network
  - MIT Raw
  - Cell
- Crippled or Mini cores
  - Intel Tflops
  - Picochip

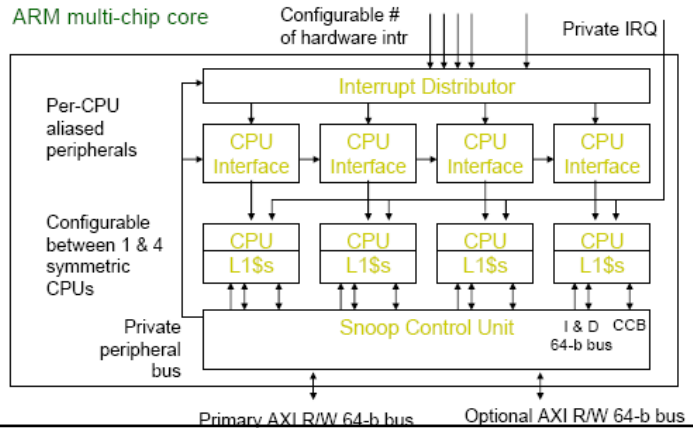
Shared Memory Multicores



- Evolution Path for Current Multicore Processors
- IBM Power5
  - Shared 1.92 Mbyte L2 cache
- AMD Opteron
  - Separate 1 Mbyte L2 caches
  - CPU0 and CPU1 communicate through the SRQ
- Intel Pentium 4
  - "Glued" two processors together

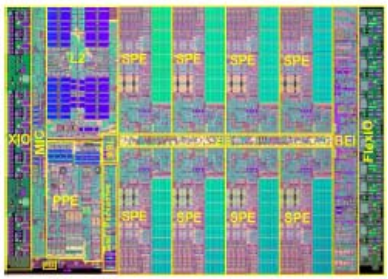
CMP

- By placing multiple processors, their memories and the IN all on one chip, the latencies of chip-to-chip communication are drastically reduced



Shared Network Multicore

- The Cell processor
- IBM/Toshiba/Sony joint project - 4-5 years, 400 designers
  - 234 million transistors, 4+ Ghz
  - 256 Gflops (billions of floating pointer operations per second)
- One 64-bit PowerPC processor
  - 4+ Ghz, dual issue, two threads
  - 512 kB of second-level cache
- Eight Synergistic Processor Elements
  - Or "Streaming Processor Elements"
  - Co-processors with dedicated 256kB of memory (not cache)
- IO
  - Dual Rambus XDR memory controllers (on chip)
  - 25.6 GB/sec of memory bandwidth
  - 76.8 GB/s chip-to-chip bandwidth (to off-chip GPU)



SMT vs CMP

- SMT exhibits better performance than CMP for network applications
  - SMT allocates hardware resources dynamically
- However:
  - more single thread processor cores can be integrated in the same die than an equivalent SMT processor
  - If an application is effectively decomposed into multiple threads it can perform better
  - Single cores are easier to design and optimize
- For network applications
  - A single-thread processor core designed to perform a specific packet processing task can be arranged in a pipelined fashion to process packets in parallel
  - Intel IXP2800 = one Xscale + 16 microengines
  - IBM PowerNP = one PowerPC + 12 picoprocessors (2 threads, 3 stage pipe)