COE 502 / CSE 661

Parallel and Vector Architectures

Prof. Muhamed Mudawar

Computer Engineering Department

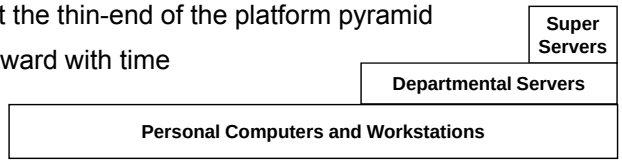
King Fahd University of Petroleum and Minerals

What will you get out of CSE 661?

- Understanding modern parallel computers
  - Technology forces
  - Fundamental architectural issues
    - Naming, replication, communication, synchronization
  - Basic design techniques
    - Pipelining
    - Cache coherence protocols
    - Interconnection networks, etc …
  - Methods of evaluation
  - Engineering tradeoffs
- From moderate to very large scale
- Across the hardware/software boundary

Will it be worthwhile?

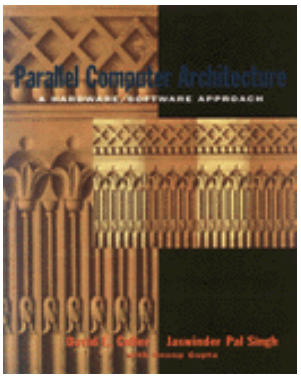
- Absolutely!
  - Even though you do not become a parallel machine designer
- Fundamental issues and solutions
  - Apply to a wide spectrum of systems
  - Crisp solutions in the context of parallel machine architecture
- Understanding implications of parallel software
- New ideas pioneered for most demanding applications
  - Appear first at the thin-end of the platform pyramid
  - Migrate downward with time



TextBook

- Parallel Computer Architecture:

  A Hardware/Software Approach
  - Culler, Singh, and Gupta
  - Morgan Kaufmann, 1999
- Covers a range of topics
- Framework & complete background
- You do the reading
- We will discuss the ideas



Research Paper Reading

- As graduate students, you are now researchers
- Most information of importance will be in research papers
- You should develop the ability to …
  - Rapidly scan and understand research papers
  - Key to your success in research
- So: you will read lots of papers in this course!
  - Students will take turns presenting and discussing papers
- Papers will be made available on the course web page

Grading Policy

- 10% Paper Readings and Presentations
- 15% Short Quizzes
- 15% Parallel Programming
- 30% Midterm Exam
- 30% Research Project
- Assignments are due at the beginning of class time

What is a Parallel Computer?

- Collection of processing elements that cooperate to solve large problems fast (Almasi and Gottlieb 1989)
- Some broad issues:
  - Resource Allocation:
    - How large a collection?
    - How powerful are the processing elements?
    - How much memory?
  - Data access, Communication and Synchronization
    - How do the elements cooperate and communicate?
    - How are data transmitted between processors?
    - What are the abstractions and primitives for cooperation?
  - Performance and Scalability
    - How does it all translate into performance?
    - How does it scale?

Why Study Parallel Architectures?

- Parallelism:
  - Provides alternative to faster clock for performance
  - Applies at all levels of system design
  - Is a fascinating perspective from which to view architecture
  - Is increasingly central in information processing
- Technological trends make parallel computing inevitable
  - Need to understand fundamental principles
- History: diverse and innovative organizational structures
  - Tied to novel programming models
- Rapidly maturing under strong technological constraints
  - Laptops and supercomputers are fundamentally similar!
  - Technological trends cause diverse approaches to converge

Role of a Computer Architect

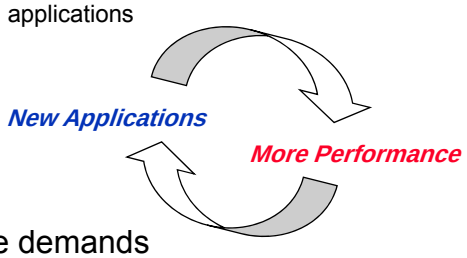
- Design and engineer various levels of a computer system
  - Understand software demands
  - Understand technology trends
  - Understand architecture trends
  - Understand economics of computer systems
- Maximize performance and programmability …
  - Within the limits of technology and cost
- Current architecture trends:
  - Today's microprocessors are multiprocessors
  - Several cores on a single chip
  - Each core capable of executing multiple threads

Is Parallel Computing Inevitable?

- Technological trends make parallel computing inevitable
- Application demands
  - Constant demand for computing cycles
  - Scientific computing, video, graphics, databases, …
- Technology Trends
  - Number of transistors on chip growing but will slow down eventually
  - Clock rates are expected to slow down (already happening!)
- Architecture Trends
  - Instruction-level parallelism valuable but limited
  - Thread-level and data-level parallelism are more promising
- Economics: Cost of pushing uniprocessor performance

Application Trends

- Application demand fuels advances in hardware
- Advances in hardware enable new applications
  - Cycle drives exponential increase in microprocessor performance
  - Drives parallel architectures
    - For most demanding applications



- Range of performance demands
  - Range of system performance with progressively increasing cost

Speedup

- A major goal of parallel computers is to achieve speedup

- Speedup (p processors) = $\dfrac{Performance(p\ processors)}{Performance(1\ processor)}$

- For a fixed problem size , *Performance* = 1 / *Time*

- Speedup fixed problem (p processors) = $\dfrac{Time(1\ processor)}{Time(p\ processors)}$

Engineering Computing Demand

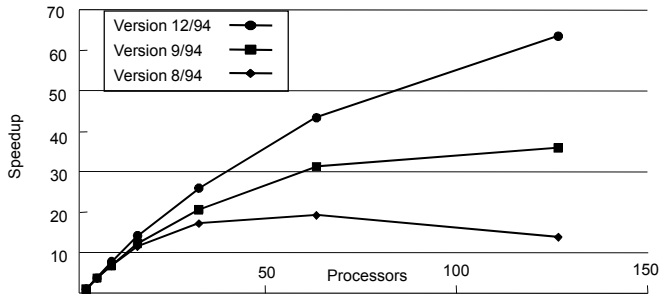
- Large parallel machines are a mainstay in many industries
  - Petroleum (reservoir analysis)
  - Aeronautics (airflow analysis, engine efficiency)
  - Computer-aided design
  - Pharmaceuticals (molecular modeling)
  - Speech and Image Processing
  - Visualization
    - In all of the above
    - Entertainment (films like Toy Story)
    - Architecture (walk-through and rendering)
  - Financial modeling (yield and derivative analysis), etc.

Commercial Computing

- Also relies on parallelism for high end
  - Large Scale servers
  - Computational power determines scale of business
- Databases, online-transaction processing, decision support, data mining, data warehousing ...
- Benchmarks
  - Explicit scaling criteria: size of database and number of users
  - Size of enterprise scales with size of system
  - Problem size increases as p increases
  - Throughput as performance measure (transactions per minute)

Improving Parallel Code

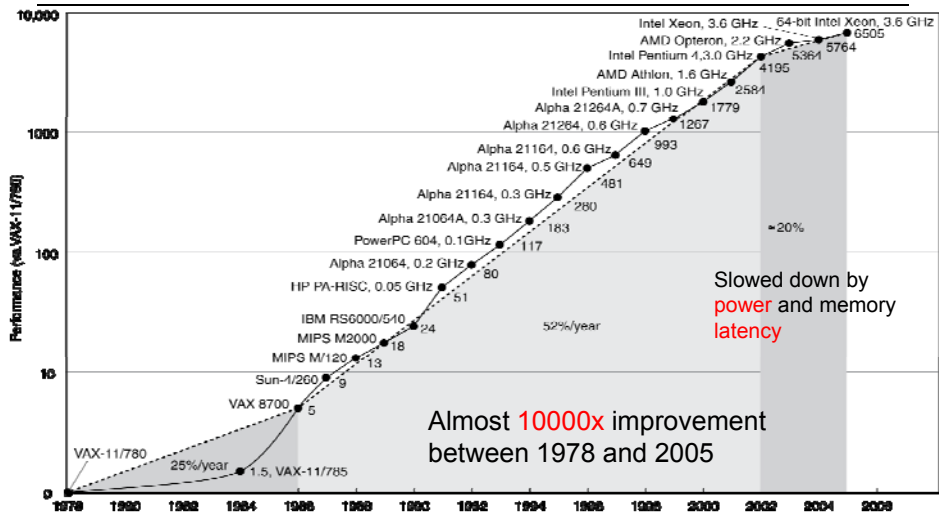
- AMBER molecular dynamics simulation program
  - Initial code was developed on Cray vector supercomputers
  - Version 8/94: good speedup for small but poor for large configurations
  - Version 9/94: improved balance of work done by each processor
  - Version 12/94: optimized communication (on Intel Paragon)



Summary of Application Trends

- Transition to parallel computing has occurred for scientific and engineering computing
- In rapid progress in commercial computing
  - Database and transactions as well as financial
  - Large-scale systems also used
- Desktop also uses multithreaded programs, which are a lot like parallel programs
- Demand for improving throughput on sequential workloads
  - Greatest use of small-scale multiprocessors
- Solid application demand exists and will increase

Uniprocessor Performance (1978-2005)



Closer Look at Processor Technology

- Basic advance is *decreasing feature size* (λ)
  - Circuits become faster
  - Die size is growing too
  - Clock rate also improves (but power dissipation is a problem)
  - Number of transistors improves like λ^2
- Performance > 100× per decade
  - Clock rate is about 10× **(no longer the case!)**
  - DRAM size quadruples every 3 years
- How to use more transistors?
  - Parallelism in processing: more functional units
    - Multiple operations per cycle reduces CPI - Clocks Per Instruction
  - Locality in data access: bigger caches
    - Avoids latency and reduces CPI, also improves processor utilization

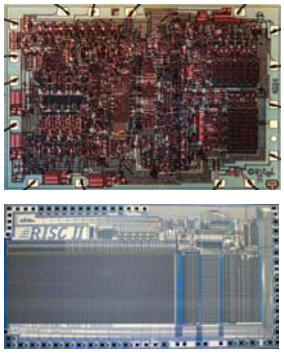
Conventional Wisdom (Patterson)

- Old Conventional Wisdom: Power is free, Transistors are expensive
- New Conventional Wisdom: "Power wall" Power is expensive, Transistors are free (Can put more on chip than can afford to turn on)
- Old CW: We can increase Instruction Level Parallelism sufficiently via compilers and innovation (Out-of-order, speculation, VLIW, …)
- New CW: "ILP wall" law of diminishing returns on more HW for ILP
- Old CW: Multiplication is slow, Memory access is fast
- New CW: "Memory wall" Memory access is slow, multiplies are fast (200 clock cycles to DRAM memory access, 4 clocks for multiply)
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall

  Uniprocessor performance now 2X / 5(?) yrs

Sea Change in Chip Design

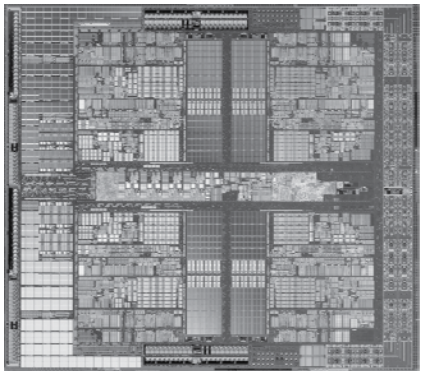
- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm^2 chip
- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm^2 chip
- 125 mm^2 chip, 65 nm CMOS = 2312 RISC II+FPU+Icache+Dcache
  - RISC II shrinks to ~ 0.02 mm^2 at 65 nm
  - New Caches and memories
- Sea change in chip design = multiple cores
  - 2X cores per chip / ~ 2 years
  - Simpler processors are more power efficient

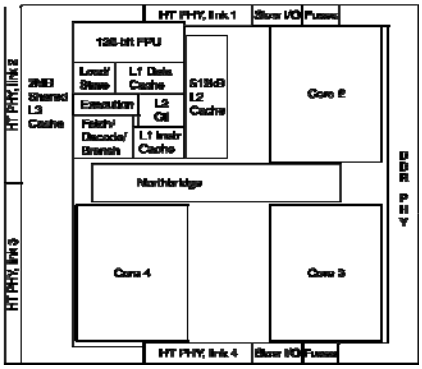


Inside a Multicore Processor Chip

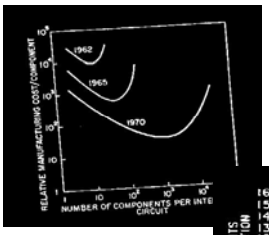
AMD Barcelona: 4 Processor **C**ores

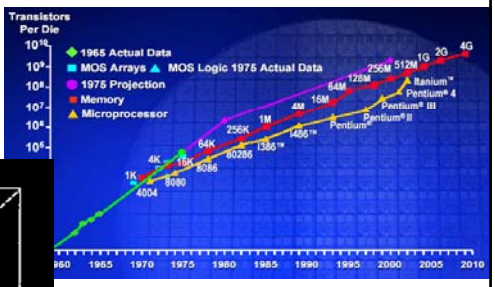
3 Levels of Caches

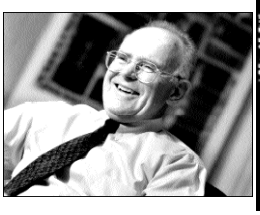


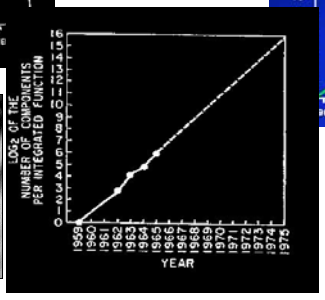


Moore's Law: 2X transistors / "year"



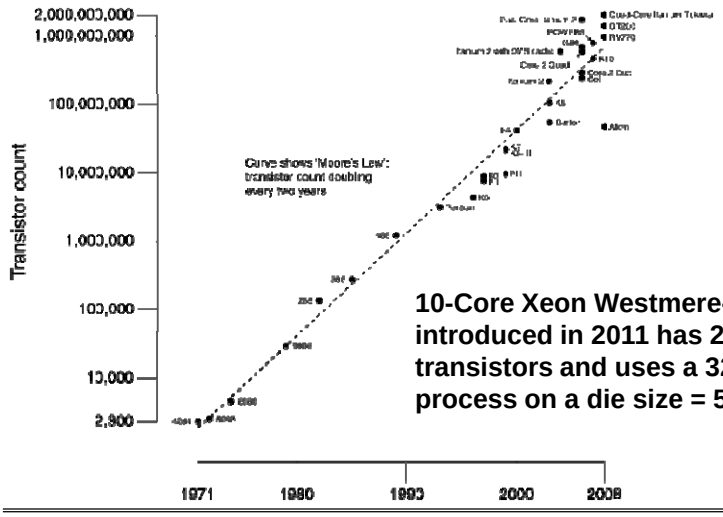






"Cramming More Components onto Integrated Circuits" Gordon Moore, Electronics, 1965
on transistors / cost-effective integrated circuit double every N months ($12 \leq N \leq 24$)

CPU Transistor Count (1971 – 2008)



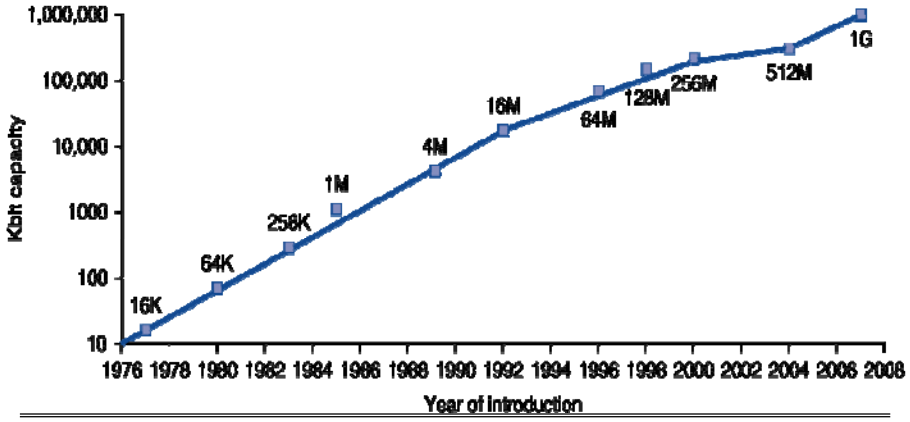
10-Core Xeon Westmere-EX introduced in 2011 has 2.6 billion transistors and uses a 32 nm process on a die size = 512 mm^2

Storage Trends

- Divergence between memory capacity and speed
  - Capacity increased by 1000x from 1980-95, speed only 2x
  - Gigabit DRAM in 2008, but gap with processor speed is widening
- Larger memories are slower, while processors get faster
  - Need to transfer more data in parallel
  - Need cache hierarchies, but how to organize caches?
- Parallelism and locality within memory systems too
  - Fetch more bits in parallel
  - Pipelined transfer of data
- Improved disk storage too
  - Using parallel disks to improve performance
  - Caching recently accessed data

Growth of Capacity per DRAM Chip

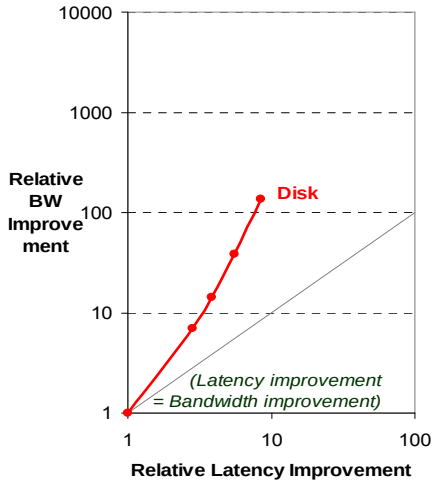
- DRAM capacity quadrupled almost every 3 years
  - 60% increase per year, for 20 years



Improvements in Disk Storage (1983 - 2003)

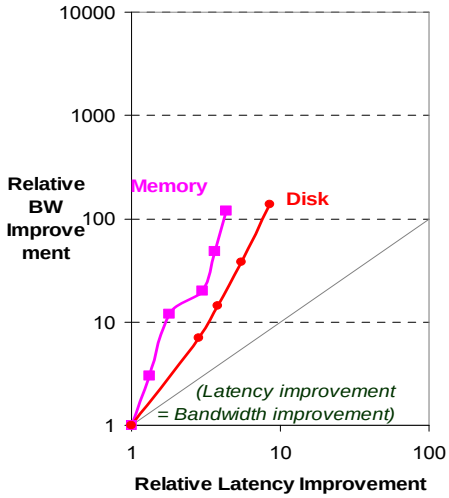
❖ CDC Wren I, 1983	❖ Seagate 373453, 2003
❖ 3600 RPM	❖ 15000 RPM (4X)
❖ 0.03 GBytes capacity	❖ 73.4 GBytes (2500X)
❖ Tracks/Inch: 800	❖ Tracks/Inch: 64000 (80X)
❖ Bits/Inch: 9550	❖ Bits/Inch: 533,000 (60X)
❖ Three 5.25” platters	❖ Four 2.5” platters (in 3.5” form factor)
❖ Bandwidth: 0.6 MBytes/sec	❖ Bandwidth: 86 MBytes/sec (143X)
❖ Latency: 48.3 ms	❖ Latency: 5.7 ms (8X)
❖ Cache: none	❖ Cache: 8 MBytes

Disk Latency vs. Bandwidth (~20 years)



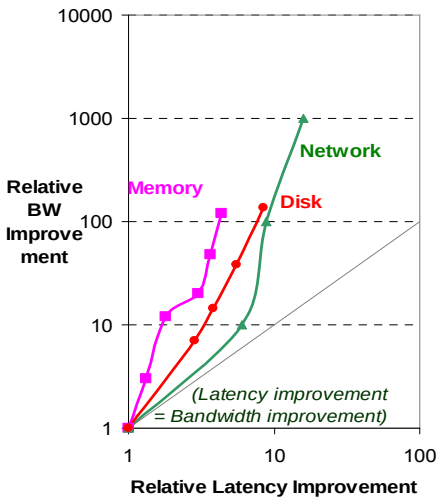
- Performance Milestones
- Disk: 3600, 5400, 7200, 10000, 15000 RPM
- Bandwidth improvement = 143X
- Latency Improvement = 8X

Memory Latency vs Bandwidth (~20 years)



- Performance Milestones
- Memory Module:
  16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM
- Bandwidth Improvement = 120X
- Latency Improvement = 4X

Network Latency vs Bandwidth (~20 years)



- Performance Milestones
- Ethernet: 10Mb/s, 100Mb/s, 1000Mb/s, 10000 Mb/s
- Bandwidth Improvement = 1000X
- Latency Improvement = 16X

Rule of Thumb for Latency Lagging Bandwidth

- In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4

  and capacity improves faster than bandwidth

- Stated alternatively:

  Bandwidth improves by more than the square of the improvement in Latency

6 Reasons Latency Lags Bandwidth

1. Moore's Law helps BW more than latency
   - Faster transistors, more transistors, more pins help Bandwidth
     - ✧ CPU Transistors: (300X in 20 years)
     - ✧ DRAM Transistors: (4000X in 20 years)
     - ✧ CPU Pins: (6X in 20 years)
     - ✧ DRAM Pins: (4X in 20 years)
   - Smaller, faster transistors but communicate over (relatively) longer lines: limits latency
     - ✧ Feature size: (17X in 20 years)
     - ✧ CPU Die Size: (6X in 20 years)
     - ✧ DRAM Die Size: (5X in 20 years)

6 Reasons Latency Lags Bandwidth (cont'd)

2. Distance increases latency
   - Size of DRAM block ⇒ long word lines and bit lines ⇒ most of DRAM access time
   - 1. & 2. explains linear latency vs. square BW
3. Bandwidth easier to sell ("bigger=better")
   - E.g., 10 Gbit/s Ethernet ("10 Gig") vs. 10 μsec latency Ethernet
   - 4400 MB/s DIMM ("PC4400") vs. 50 ns latency
   - Even if just marketing, customers now trained
   - Since bandwidth sells, more resources thrown at bandwidth, which further tips the balance

6 Reasons Latency Lags Bandwidth (cont'd)

4. Latency helps BW, but not vice versa
   - Spinning disk faster improves both rotational latency and bandwidth
     - 3600 RPM ⇒ 15000 RPM = 4.2X
     - Average rotational latency: 8.3 ms ⇒ 2.0 ms
     - Things being equal, also helps BW by 4.2X
   - Lower DRAM latency ⇒ More DRAM access/second (higher bandwidth)
   - Higher linear density helps disk BW (and capacity), but not disk Latency
     - 9,550 BPI ⇒ 533,000 BPI ⇒ 60X in BW, but not in latency

6 Reasons Latency Lags Bandwidth (cont'd)

5. Bandwidth hurts latency
   - Adding chips to widen a memory module increases Bandwidth but higher fan-out on address lines may increase Latency
6. Operating System overhead hurts Latency more than Bandwidth
   - Scheduling queues increase latency (more delays)
   - Queues help Bandwidth, but hurt Latency

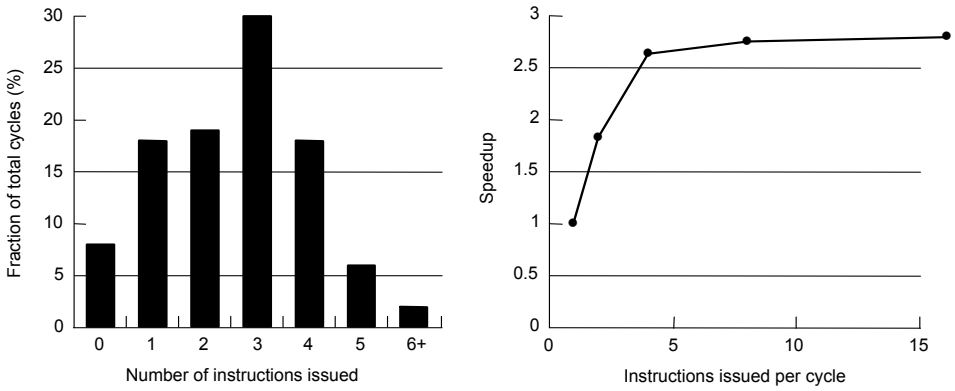
Architectural Trends

- Architecture translates technology gifts into performance and capability
- Resolves the tradeoff between parallelism and locality
  - Current microprocessor: 1/3 compute, 1/3 cache, 1/3 off-chip connect
  - Tradeoffs may change with scale and technology advances
- Understanding microprocessor architectural trends
  - Helps build intuition about design issues or parallel machines
  - Shows fundamental role of parallelism even in “sequential” computers
- Four generations: tube, transistor, IC, VLSI
  - Here focus only on VLSI generation
- Greatest trend in VLSI has been in type of parallelism exploited

Architecture: Increase in Parallelism

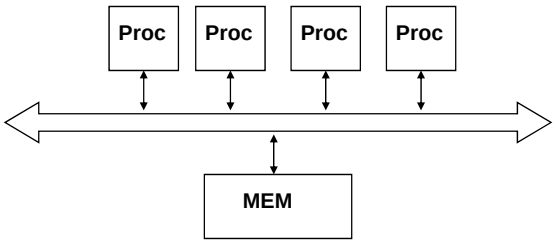
- Bit level parallelism (before 1985) 4-bit → 8-bit → 16-bit
  - Slows after 32-bit processors
  - Adoption of 64-bit in late 90s, 128-bit and beyond for vector processing
  - Great inflection point when 32-bit processor and cache fit on a chip
- Instruction Level Parallelism (ILP): Mid 80s until late 90s
  - Pipelining and simple instruction sets (RISC) + compiler advances
  - On-chip caches and functional units => superscalar execution
  - Greater sophistication: out of order execution and hardware speculation
- Today: thread level parallelism and chip multiprocessors
  - Thread level parallelism goes beyond instruction level parallelism
  - Running multiple threads in parallel inside a processor chip
  - Fitting multiple processors and their interconnect on a single chip

How far will ILP go?



Limited ILP under ideal superscalar execution: infinite resources and fetch bandwidth, perfect branch prediction and renaming, but real cache. At most 4 instruction issue per cycle 90% of the time.

Thread-Level Parallelism "on board"



- Microprocessor is a building block for a multiprocessor
  - Makes it natural to connect many to shared memory
- Faster processors saturate bus
  - Interconnection networks are used in larger scale systems

No. of processors in fully configured commercial shared-memory systems

Supercomputing Trends

- Quest to achieve absolute maximum performance
- Supercomputing has historically been proving ground and a driving force for innovative architectures and techniques
- Very small market
- Dominated by vector machines in the 70s
  - Vector operations permit data parallelism within a single thread
  - Vector processors were implemented in fast, high-power circuit technologies in small quantities which made them very expensive
- Multiprocessors now replace vector supercomputers
  - Microprocessors have made huge gains in clock rates, floating-point performance, pipelined execution, instruction-level parallelism, effective use of caches, and large volumes

Summary: Why Parallel Architectures

- Increasingly attractive
  - Economics, technology, architecture, application demand
- Increasingly central and mainstream
- Parallelism exploited at many levels
  - Instruction-level parallelism
  - Thread-level parallelism
  - Data-level parallelism

  } **Our Focus in this course**
- Same story from memory system perspective
  - Increase bandwidth, reduce average latency with local memories
- Spectrum of parallel architectures make sense
  - Different cost, performance, and scalability