ECE 669

Parallel Computer Architecture

Lecture 27

Course Wrap Up

What is Parallel Architecture?

- A *parallel computer* is a collection of processing elements that cooperate to solve large problems fast

- Some broad issues:
  - Resource Allocation:
    - how large a collection?
    - how powerful are the 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 Architecture?

Role of a computer architect:

To design and engineer the various levels of a computer system to maximize *performance* and *programmability* within limits of *technology* and *cost*.

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

Speedup

- Speedup (p processors) =

$$\frac{\textit{Performance (p processors)}}{\textit{Performance (1 processor)}}$$

- For a fixed problem size (input data set), performance = 1/time

- Speedup fixed problem (p processors) =

$$\frac{\textit{Time (1 processor)}}{\textit{Time (p processors)}}$$

Architectural Trends

- Architecture translates technology’s 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

Architectural Trends

° **Greatest trend in VLSI generation is increase in parallelism**
  - **Up to 1985: bit level parallelism: 4-bit -> 8 bit -> 16-bit**
    - **slows after 32 bit**
    - **adoption of 64-bit now under way, 128-bit far (not performance issue)**
    - **great inflection point when 32-bit micro and cache fit on a chip**
  - **Mid 80s to mid 90s: instruction level parallelism**
    - **pipelining and simple instruction sets, + compiler advances (RISC)**
    - **on-chip caches and functional units => superscalar execution**
    - **greater sophistication: out of order execution, speculation, prediction**
      - to deal with control transfer and latency problems
  - **Next step: thread level parallelism**

Summary: Why Parallel Architecture?

- **Increasingly attractive**
  - Economics, technology, architecture, application demand
- **Increasingly central and mainstream**
- **Parallelism exploited at many levels**
  - Instruction-level parallelism
  - Multiprocessor servers
  - Large-scale multiprocessors ("MPPs")
- **Focus of this class: multiprocessor level of parallelism**
- **Same story from memory system perspective**
  - Increase bandwidth, reduce average latency with many local memories
- **Spectrum of parallel architectures make sense**
  - Different cost, performance and scalability

Programming Model

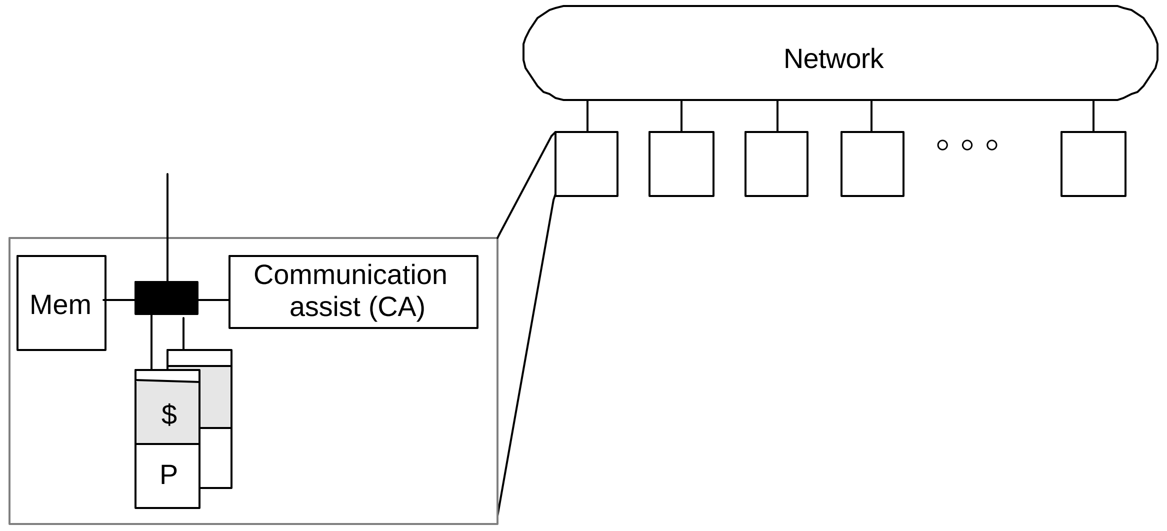
- ***Conceptualization of the machine that programmer uses in coding applications***
  - **How parts cooperate and coordinate their activities**
  - **Specifies communication and synchronization operations**

- **Multiprogramming**
  - **no communication or synch. at program level**
- ***Shared address space***
  - **like bulletin board**
- ***Message passing***
  - **like letters or phone calls, explicit point to point**
- ***Data parallel*:**
  - **more regimented, global actions on data**
  - **Implemented with shared address space or message passing**

Toward Architectural Convergence

- Evolution and role of software have blurred boundary
    - Send/recv supported on SAS machines via buffers
- Hardware organization converging too
    - Tighter NI integration even for MP (low-latency, high-bandwidth)
    - Hardware SAS passes messages
- Even clusters of workstations/SMPs are parallel systems
    - Emergence of fast system area networks (SAN)
- Programming models distinct, but organizations converging
    - Nodes connected by general network and communication assists
    - Implementations also converging, at least in high-end machines

Convergence: Generic Parallel Architecture



- Node: processor(s), memory system, plus *communication assist*
  - Network interface and communication controller
- Scalable network
- Convergence allows lots of innovation, within framework
  - Integration of assist with node, what operations, how efficiently...

Architecture

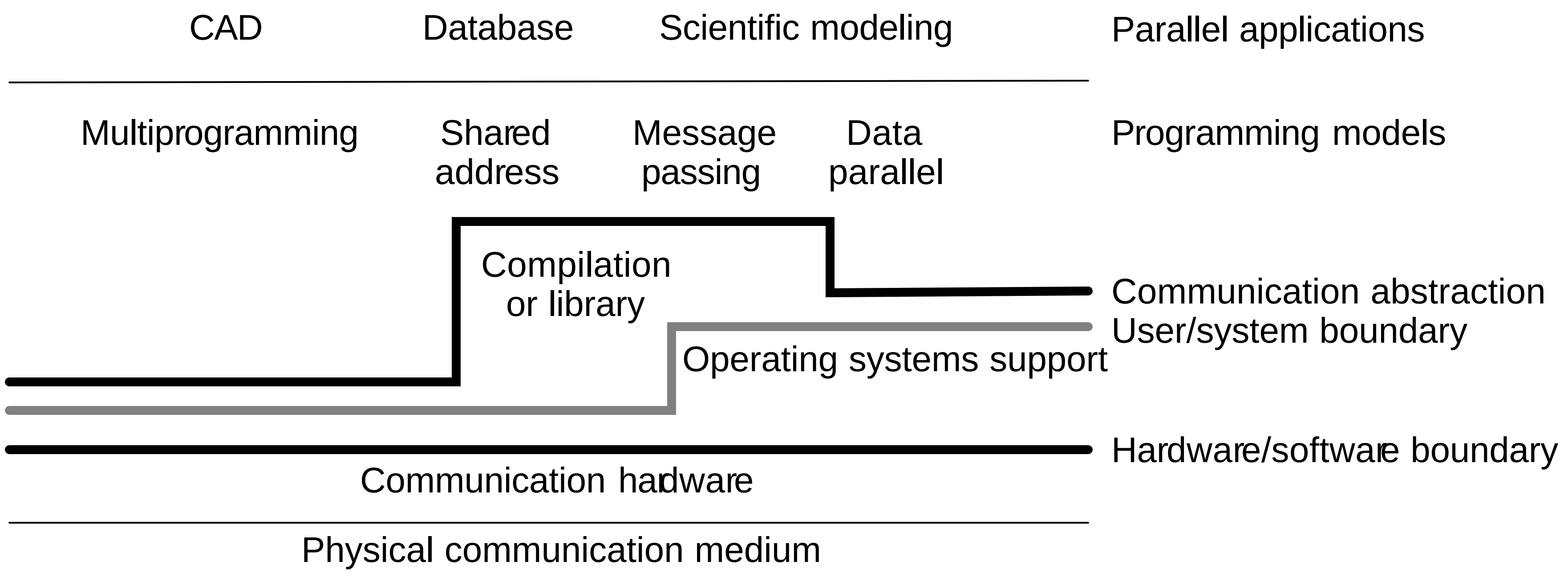
- Two facets of Computer Architecture:
  - Defines Critical Abstractions
    - especially at HW/SW boundary
    - set of operations and data types these operate on
  - Organizational structure that realizes these abstraction
- Parallel Computer Arch. =
  Comp. Arch + Communication Arch.
- Comm. Architecture has same two facets
  - communication abstraction
  - primitives at user/system and hw/sw boundary

Communication Architecture

User/System Interface + Organization

- **User/System Interface:**
  - **Comm. primitives exposed to user-level by hw and system-level sw**
- **Implementation:**
  - **Organizational structures that implement the primitives: HW or OS**
  - **How optimized are they? How integrated into processing node?**
  - **Structure of network**
- **Goals:**
  - **Performance**
  - **Broad applicability**
  - **Programmability**
  - **Scalability**
  - **Low Cost**

Modern Layered Framework



Understanding Parallel Architecture

- Traditional taxonomies not very useful
- Programming models not enough, nor hardware structures
  - Same one can be supported by radically different architectures

=> Architectural distinctions that affect software

  - Compilers, libraries, programs

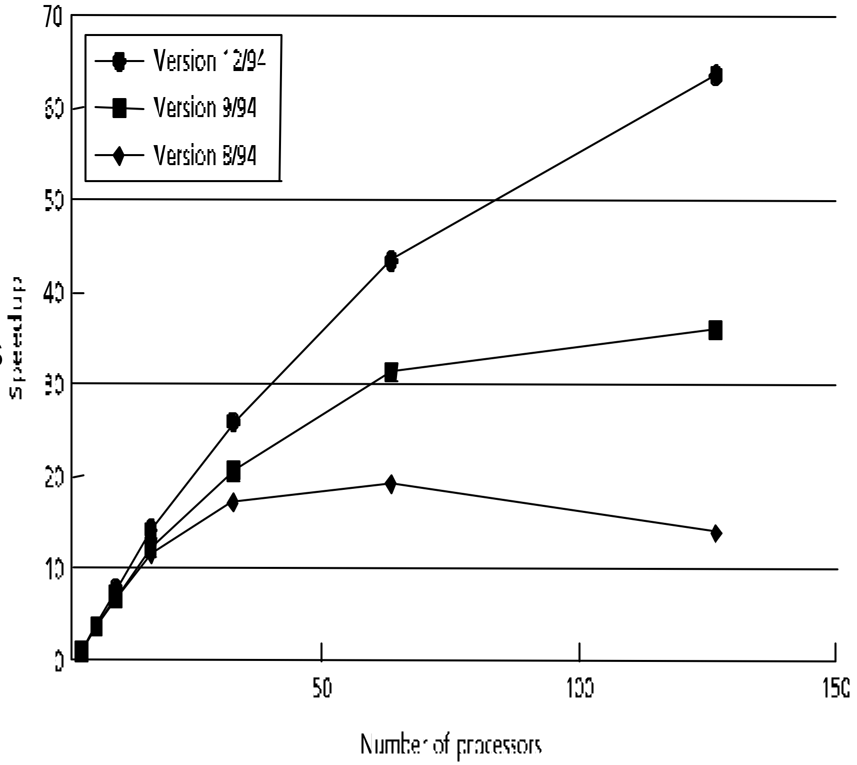
- Design of user/system and hardware/software interface
  - Constrained from above by progr. models and below by technology
- Guiding principles provided by layers
  - What primitives are provided at communication abstraction
  - How programming models map to these
  - How they are mapped to hardware

Fundamental Design Issues

- **At any layer, interface (contract) aspect and performance aspects**
  - **Naming: How are logically shared data and/or processes referenced?**
  - **Operations: What operations are provided on these data**
  - **Ordering: How are accesses to data ordered and coordinated?**
  - **Replication: How are data replicated to reduce communication?**
  - **Communication Cost: Latency, bandwidth, overhead, occupancy**

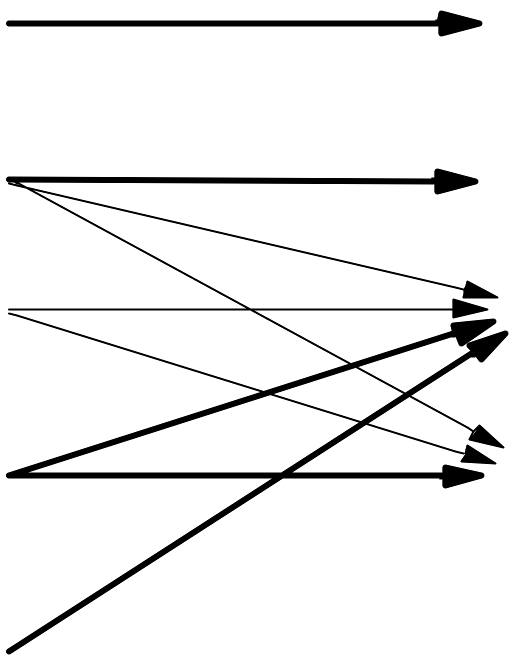
Performance Goal => Speedup

- Architect Goal
  - observe how program uses machine and improve the design to enhance performance
- Programmer Goal
  - observe how the program uses the machine and improve the implementation to enhance performance
- What do you observe?
- Who fixes what?



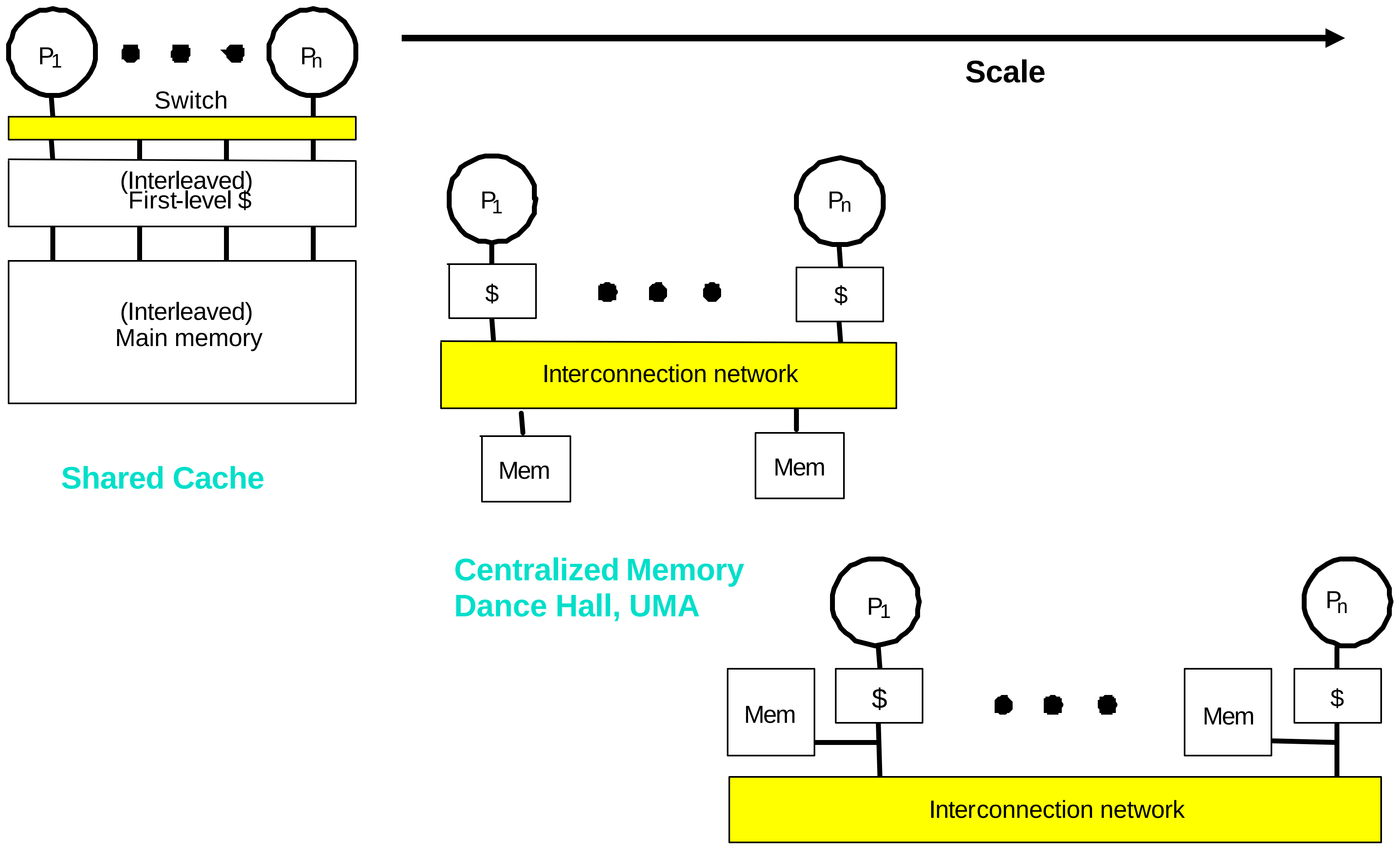
Relationship between Perspectives

Parallelization step(s)	Performance issue	Processor time component
Decomposition/ assignment/ orchestration	Load imbalance and synchronization	Synch wait
Decomposition/ assignment	Extra work	Busy-overhead
Decomposition/ assignment	Inherent communication volume	Data-remote
Orchestration	Artifactual communication and data locality	Data-local
Orchestration/ mapping	Communication structure	



Speedup ≤ $$\frac{Busy(1) + Data(1)}{Busy_{useful}(p)+Data_{local}(p)+Synch(p)+Data_{remote}(p)+Busy_{overhead}(p)}$$

Natural Extensions of Memory System



Shared Cache

Centralized Memory
Dance Hall, UMA

Distributed Memory (NUMA)

Workload-Driven Evaluation

- Evaluating real machines
- Evaluating an architectural idea or trade-offs

=> need good metrics of performance

=> need to pick good workloads

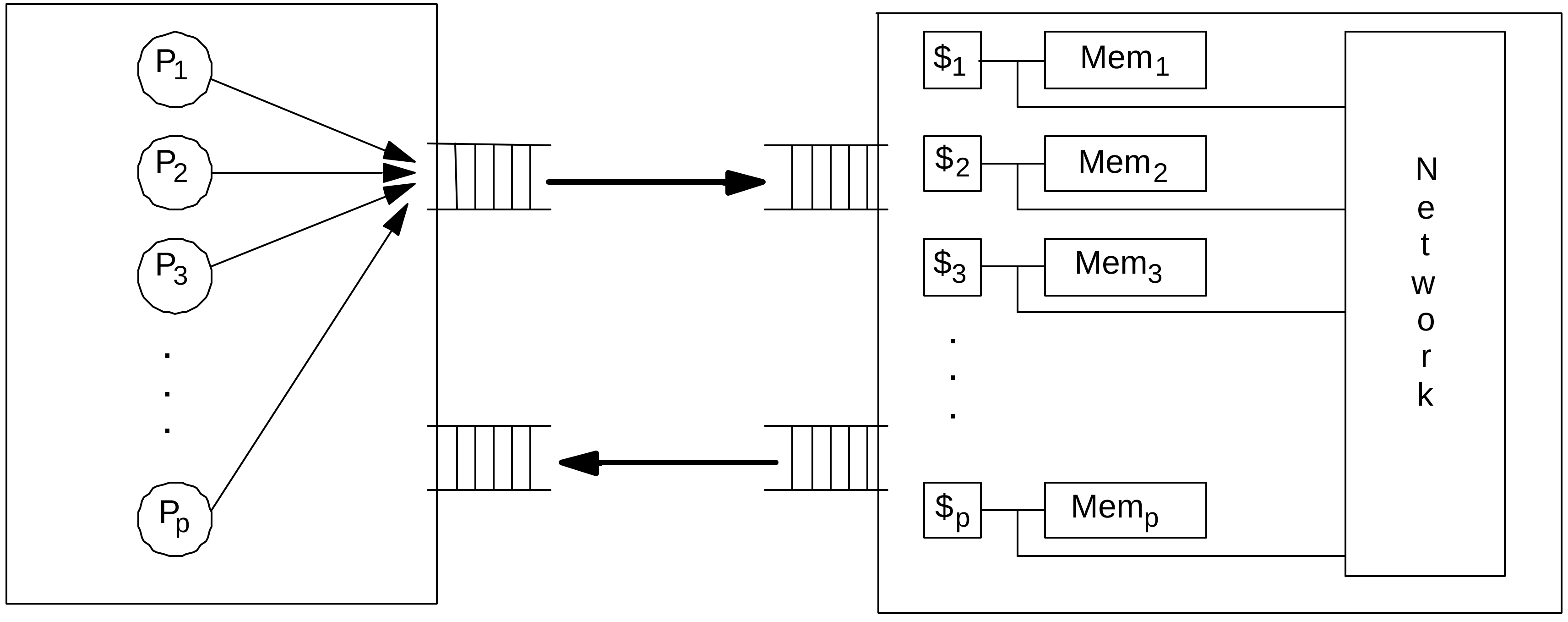
=> need to pay attention to scaling

- many factors involved

- Today: narrow architectural comparison
- Set in wider context

Execution-driven Simulation

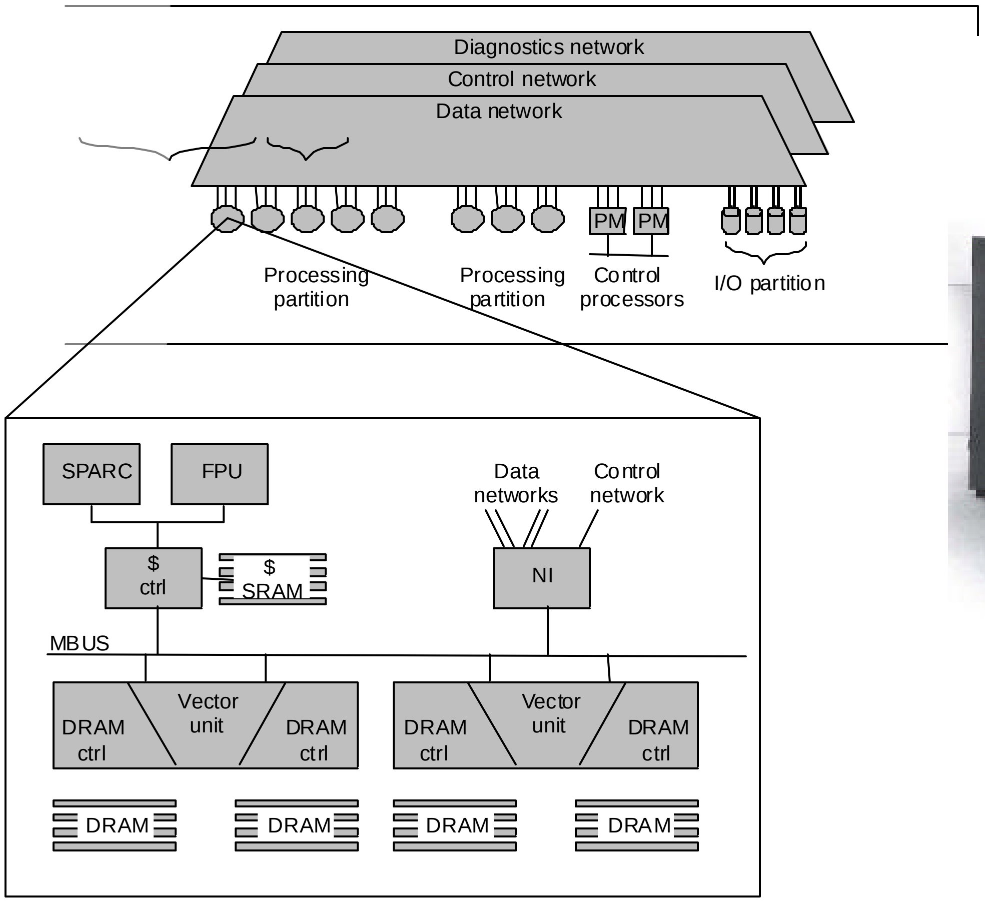
- Memory hierarchy simulator returns simulated time information to reference generator, which is used to schedule simulated processes



Reference generator

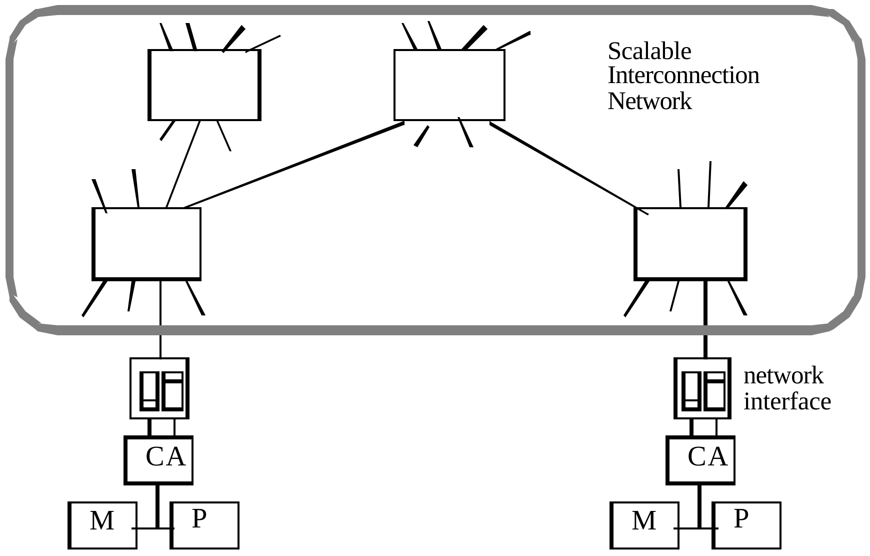
Memory and interconnect simulator

CM-5 Machine Organization



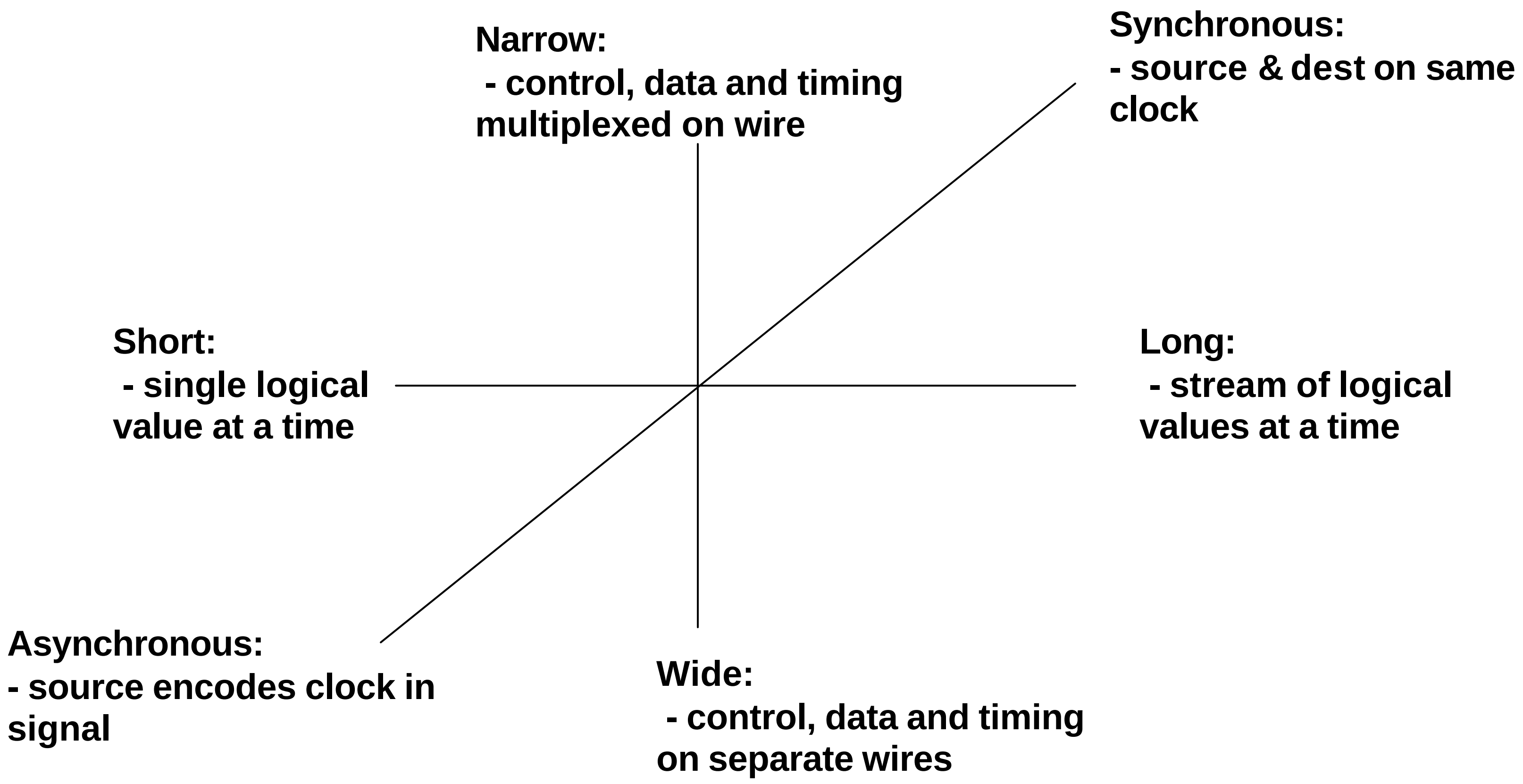
Scalable, High Performance Interconnection Network

- At Core of Parallel Computer Arch.
- Requirements and trade-offs at many levels
  - Elegant mathematical structure
  - Deep relationships to algorithm structure
  - Managing many traffic flows
  - Electrical / Optical link properties
- Little consensus
  - interactions across levels
  - Performance metrics?
  - Cost metrics?
  - Workload?



Link Design/Engineering Space

- Cable of one or more wires/fibers with connectors at the ends attached to switches or interfaces


Narrow:
- control, data and timing multiplexed on wire
Synchronous:
- source & dest on same clock
Short:
- single logical value at a time
Long:
- stream of logical values at a time
Asynchronous:
- source encodes clock in signal
Wide:
- control, data and timing on separate wires

Routing

- Routing Algorithms restrict the set of routes within the topology
  - simple mechanism selects turn at each hop
  - arithmetic, selection, lookup
- Deadlock-free if channel dependence graph is acyclic
  - limit turns to eliminate dependences
  - add separate channel resources to break dependences
  - combination of topology, algorithm, and switch design
- Deterministic vs adaptive routing
- Switch design issues
  - input/output buffering, routing logic, selection logic
- Flow control
- Real networks are a ‘package’ of design choices

Looking Forward

- The only constant is “constant change”
- Where will the next “1000x” come from?
  - it is likely to be driven by the narrow top of the platform pyramid serving the most demanding applications
  - it will be constructed out of the technology and building blocks of the very large volume
  - it will be driven by billions of people utilizing ‘infrastructure services’

Prognosis

- **Continuing on current trends, reach a petaop/s in 2010**
  - **clock rate is tiny fraction, density dominates**
  - **translating area into performance is PARALLEL ARCH**
- **Better communication interface**
  - **10 GB/s links are on the drawing board**
  - **NGIO/FutureIO will standardize port into memory controller**
- **Gap to DRAM will grow, and grow, and grow...**
  - **processors will become more latency tolerant**
  - **many instructions per thread with OO exec**
  - **many threads'**
- **Bandwidth is key**
- **Proc diminishing fraction of chip**
  - **and unfathomably complex**

Continuing Out

- **Proc and Memory will integrate on chip**
  - **everything beyond embedded devices will be MP**
  - **PIN = Communication**
- **Systems will be a federation of components on a network**
  - **every component has a processor inside**
    - **disk, display, microphone, ...**
  - **every system is a parallel machine**
  - **how do we make them so that they just work?**

Fundamental Limits?

- **Speed of light**
  - **Latency dominated by occupancy**
  - **occupancy dominated by overhead**
    - **its all in the connectors**
  - **communication performance fundamentally limited by design methodology**
    - **make the local case fast at the expense of the infrequent remote case**
  - **this may change when we a fundamentally managing information flows, rather than managing state**
    - **we're seeing this change at many levels**

What Turing Said

[Gray 5/99]

“I believe that in about fifty years' time it will be possible, to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent chance of making the right identification after five minutes of questioning. The original question, "Can machines think?" I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.”

Alan M.Turing, 1950

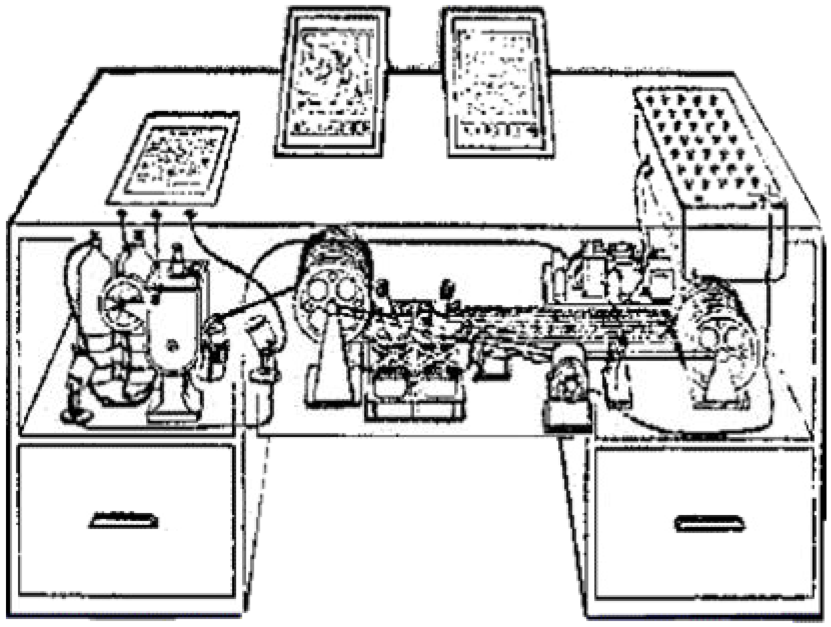
“Computing machinery and intelligence.” *Mind*, Vol. LIX. 433-460

Vannevar Bush (1890-1974)

"As We May Think" *The Atlantic Monthly,* July 1945
http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

[Gray 5/99]

- **Memex**
  All human knowledge
  in Memex
  "a billion books"
  hyper-linked together
- Record everything you see
  - camera glasses
  - "a machine which types when talked to"
- Navigate by
  text search
  following links
  associations.



- **Direct electrical path to human nervous system?**

Memex is Here! (or near)

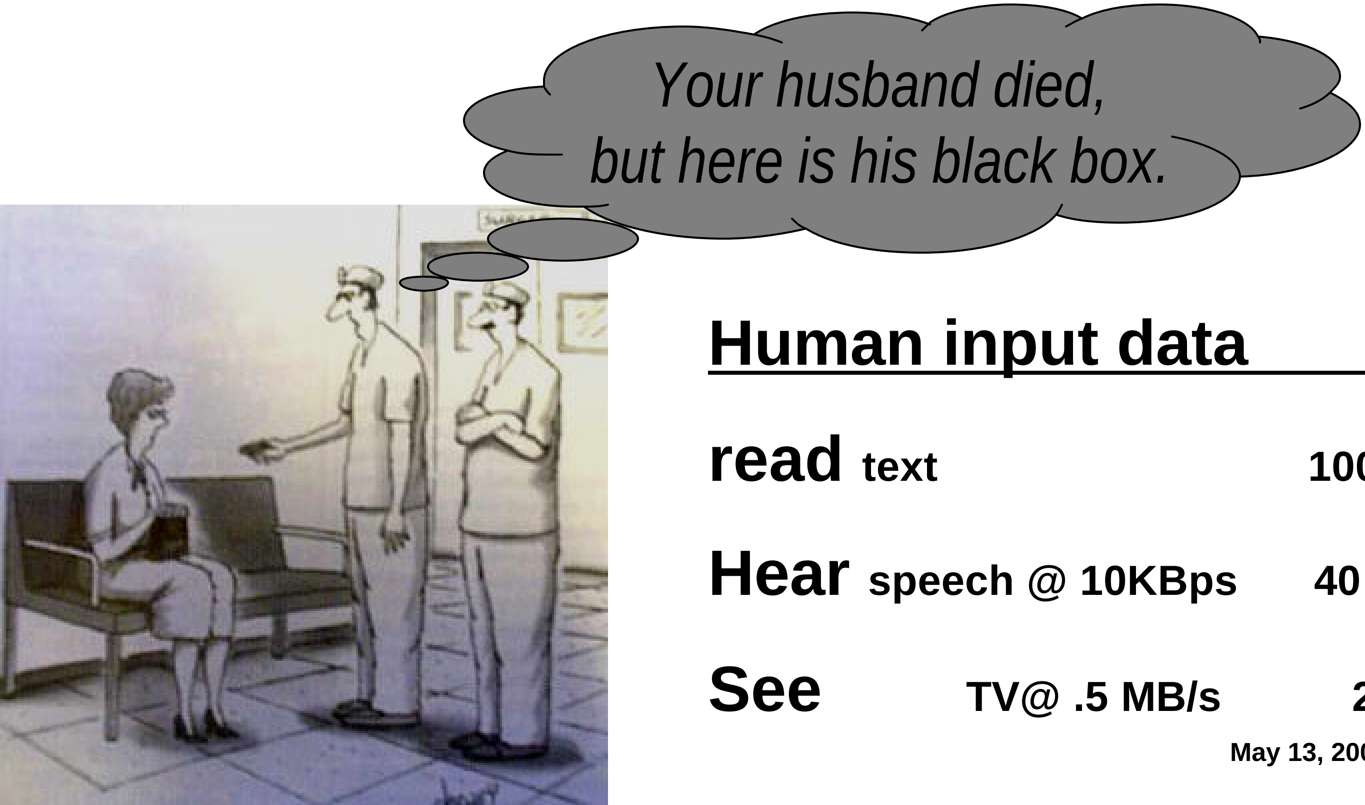
[Gray 5/99]

- The Internet is growing fast.
- Most scientific literature is online somewhere.
  - it doubles every 10 years!
- Most literature is online (but copyrighted).
- Most Library of Congress visitors: web.
- A problem Bush anticipated:
  Finding answers is hard.

Personal Memex

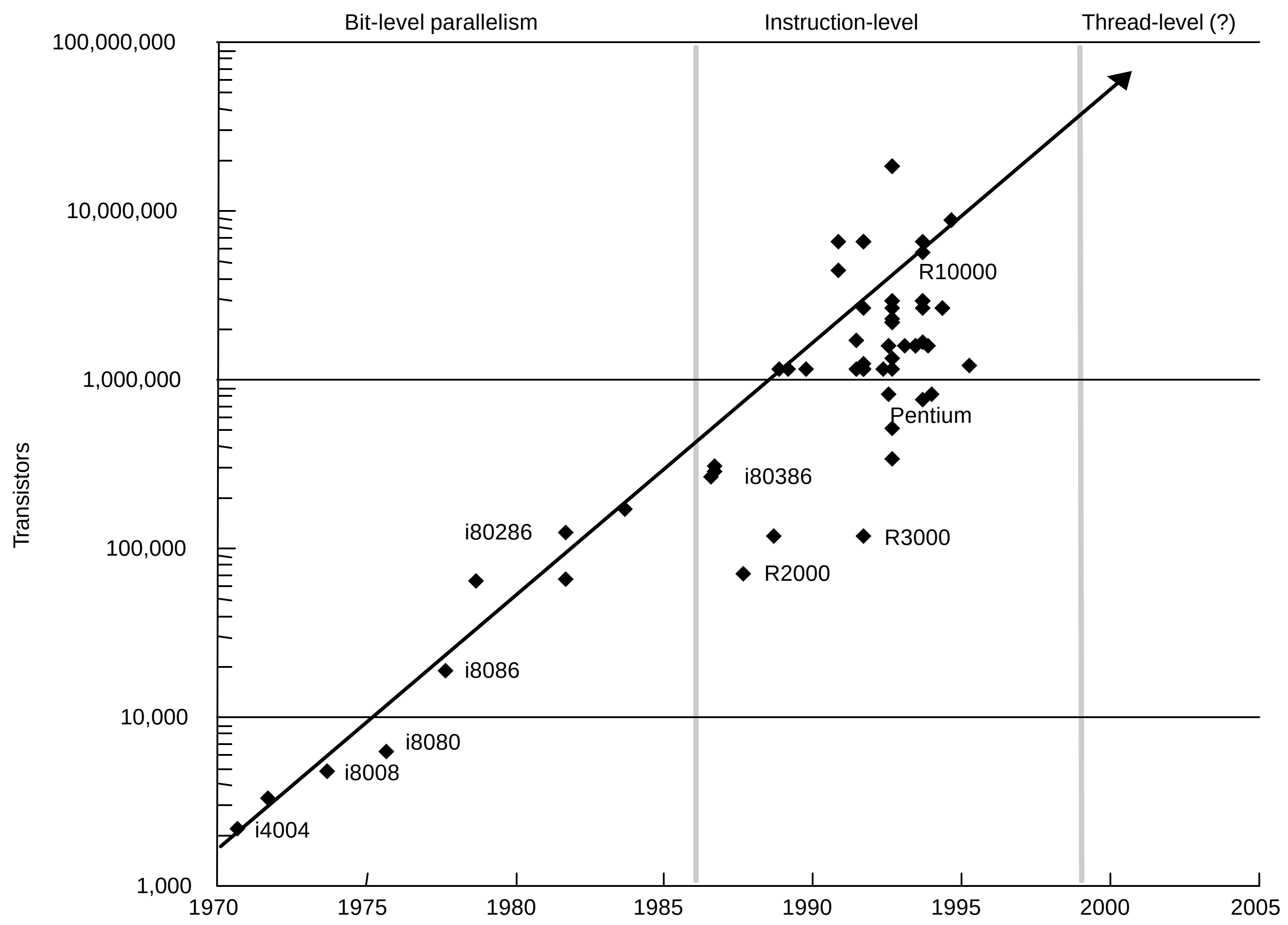
[Gray 5/99]

- Remember what is seen and heard and quickly return any item on request.



Human input data	/hr	/lifetime
read text	100 KB	25 GB
Hear speech @ 10KBps	40 MB	10 TB
See TV@ .5 MB/s	2 GB	8 PB

Phases in “VLSI” Generation



Parallel Architecture Summary

- Lots of new innovation left
  - New technologies
  - New communication techniques
  - New compilation techniques
- Current technology migrating towards common platform/compilation environment
  - Ties with distributed computing
  - Ties with global computing
- Lots of hard problems left to solve
  - Military, commercial, weather, astronomy, etc
- Improving the human interface
  - Do I really need a keyboard?