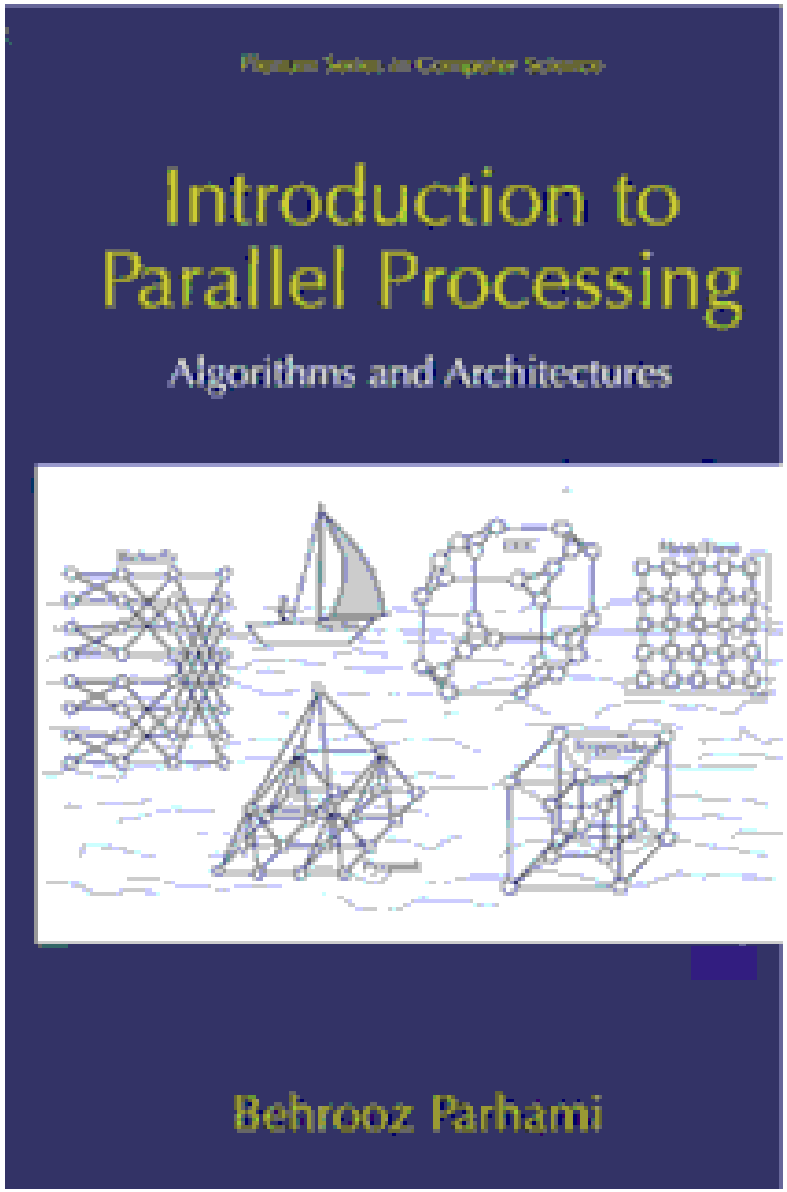



# Part VI
## Implementation Aspects

Architectural Variations

| | | |
|---|---|---|
| Part I: Fundamental Concepts | Background and Motivation | 1. Introduction to Parallelism |
| | | 2. A Taste of Parallel Algorithms |
| | Complexity and Models | 3. Parallel Algorithm Complexity |
| | | 4. Models of Parallel Processing |
| Part II: Extreme Models | Abstract View of Shared Memory | 5. PRAM and Basic Algorithms |
| | | 6. More Shared-Memory Algorithms |
| | Circuit Model of Parallel Systems | 7. Sorting and Selection Networks |
| | | 8. Other Circuit-Level Examples |
| Part III: Mesh-Based Architectures | Data Movement on 2D Arrays | 9. Sorting on a 2D Mesh or Torus |
| | | 10. Routing on a 2D Mesh or Torus |
| | Mesh Algorithms and Variants | 11. Numerical 2D Mesh Algorithms |
| | | 12. Other Mesh-Related Architectures |
| Part IV: Low-Diameter Architectures | The Hypercube Architecture | 13. Hypercubes and Their Algorithms |
| | | 14. Sorting and Routing on Hypercubes |
| | Hypercubic and Other Networks | 15. Other Hypercubic Architectures |
| | | 16. A Sampler of Other Networks |
| Part V: Some Broad Topics | Coordination and Data Access | 17. Emulation and Scheduling |
| | | 18. Data Storage, Input, and Output |
| | Robustness and Ease of Use | 19. Reliable Parallel Processing |
| | | 20. System and Software Issues |
| Part VI: Implementation Aspects | Control-Parallel Systems | 21. Shared-Memory MIMD Machines |
| | | 22. Message-Passing MIMD Machines |
| | Data Parallelism and Conclusion | 23. Data-Parallel SIMD Machines |
| | | 24. Past, Present, and Future |

# About This Presentation

This presentation is intended to support the use of the textbook *Introduction to Parallel Processing: Algorithms and Architectures* (Plenum Press, 1999, ISBN 0-306-45970-1). It was prepared by the author in connection with teaching the graduate-level course ECE 254B: Advanced Computer Architecture: Parallel Processing, at the University of California, Santa Barbara. Instructors can use these slides in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

| Edition | Released | Revised | Revised |
|---|---|---|---|
| First | Spring 2005 | Fall 2008* | |

* Very limited update

# About This Presentation

This presentation is intended to support the use of the textbook *Introduction to Parallel Processing: Algorithms and Architectures* (Plenum Press, 1999, ISBN 0-306-45970-1). It was prepared by the author in connection with teaching the graduate-level course ECE 254B: Advanced Computer Architecture: Parallel Processing, at the University of California, Santa Barbara. Instructors can use these slides in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

| Edition | Released | Revised | Revised | Revised |
|---|---|---|---|---|
| First | Spring 2005 | Fall 2008 | Fall 2010* | |
| | | | | |
| | | | | |

* Very limited update

# VI Implementation Aspects

Study real parallel machines in MIMD and SIMD classes:

- Examine parallel computers of historical significance
- Learn from modern systems and implementation ideas
- Bracket our knowledge with history and forecasts

| Topics in This Part | |
|---|---|
| Chapter 21 | Shared-Memory MIMD Machines |
| Chapter 22 | Message-Passing MIMD Machines |
| Chapter 23 | Data-Parallel SIMD Machines |
| Chapter 24 | Past, Present, and Future |

# 21 Shared-Memory MIMD Machines

Learn about practical shared-variable parallel architectures:

- Contrast centralized and distributed shared memories
- Examine research prototypes and production machines

| Topics in This Chapter | |
|---|---|
| 21.1 | Variations in Shared Memory |
| 21.2 | MIN-Based BBN Butterfly |
| 21.3 | Vector-Parallel Cray Y-MP |
| 21.4 | Latency-Tolerant Tera MTA |
| 21.5 | CC-NUMA Stanford DASH |
| 21.6 | SCI-Based Sequent NUMA-Q |

# 21.1 Variations in Shared Memory

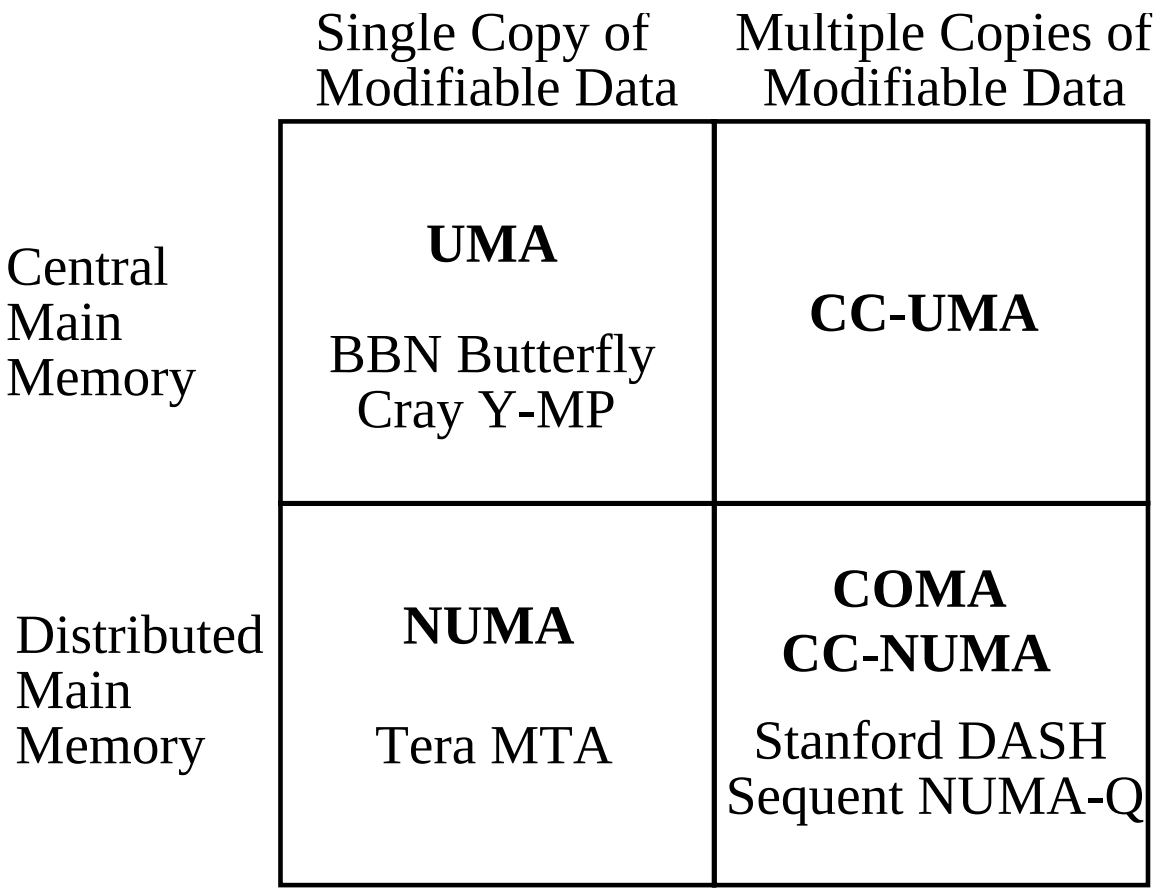



Fig. 21.1 Classification of shared-memory hardware architectures and example systems that will be studied in the rest of this chapter.

# C.mmp: A Multiprocessor of Historical Significance

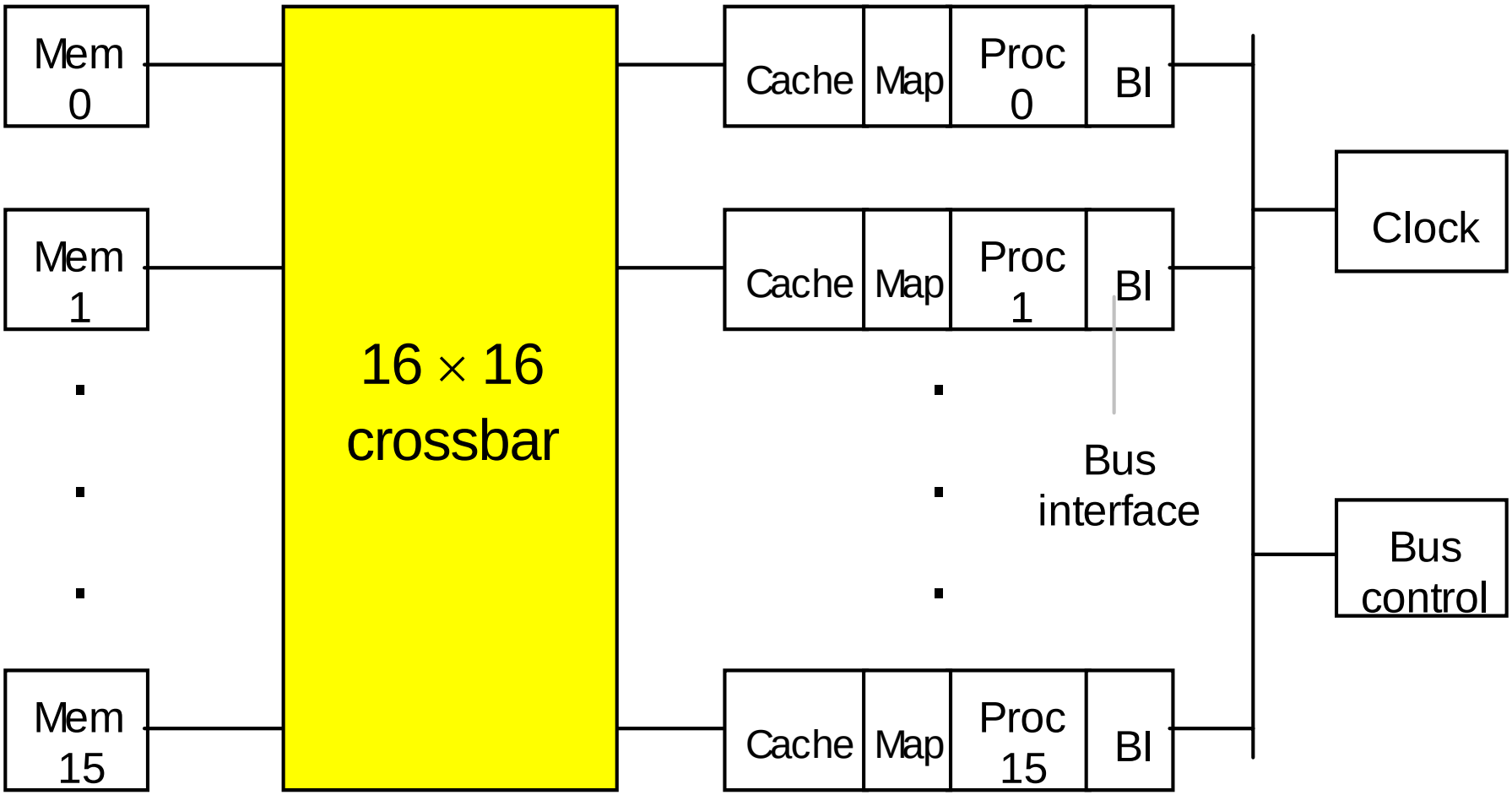



Fig. 21.2 Organization of the C.mmp multiprocessor.

# Shared Memory Consistency Models

Varying latencies makes each processor's view of the memory different

**Sequential consistency** (strictest and most intuitive); mandates that the interleaving of reads and writes be the same from the viewpoint of all processors. This provides the illusion of a FCFS single-port memory.

**Processor consistency** (laxer); only mandates that writes be observed in the same order by all processors. This allows reads to overtake writes, providing better performance due to out-of-order execution.

**Weak consistency** separates ordinary memory accesses from synch accesses that require memory to become consistent. Ordinary read and write accesses can proceed as long as there is no pending synch access, but the latter must wait for all preceding accesses to complete.

**Release consistency** is similar to weak consistency, but recognizes two synch accesses, called "acquire" and "release", that sandwich protected shared accesses. Ordinary read/write accesses can proceed only when there is no pending acquire access from the same processor and a release access must wait for all reads and writes to be completed.

# 21.2 MIN-Based BBN Butterfly

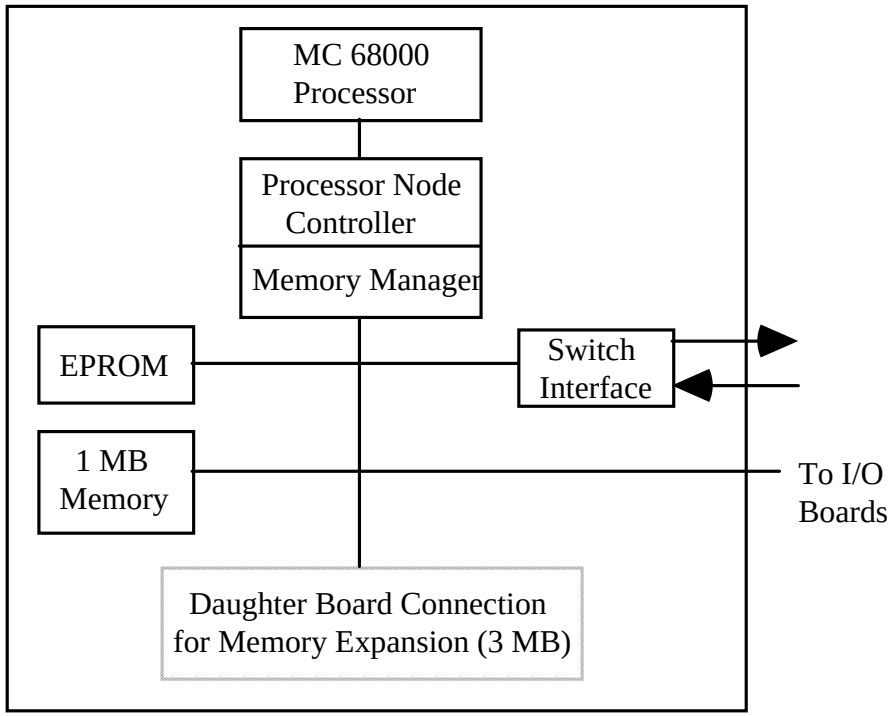



Fig. 21.3 Structure of a processing node in the BBN Butterfly.

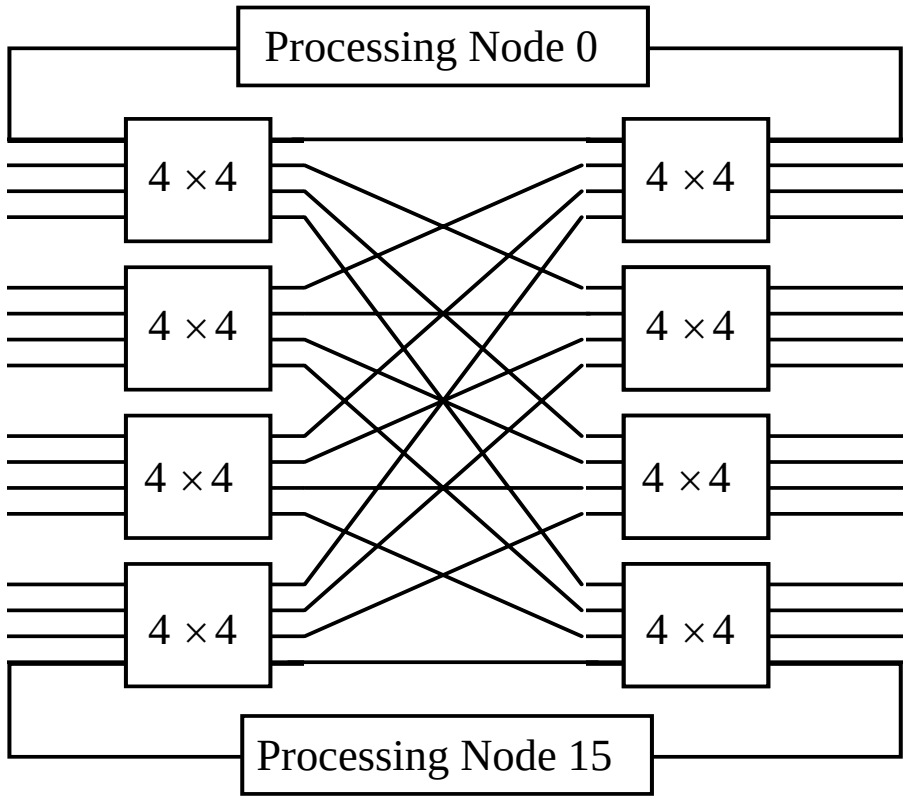
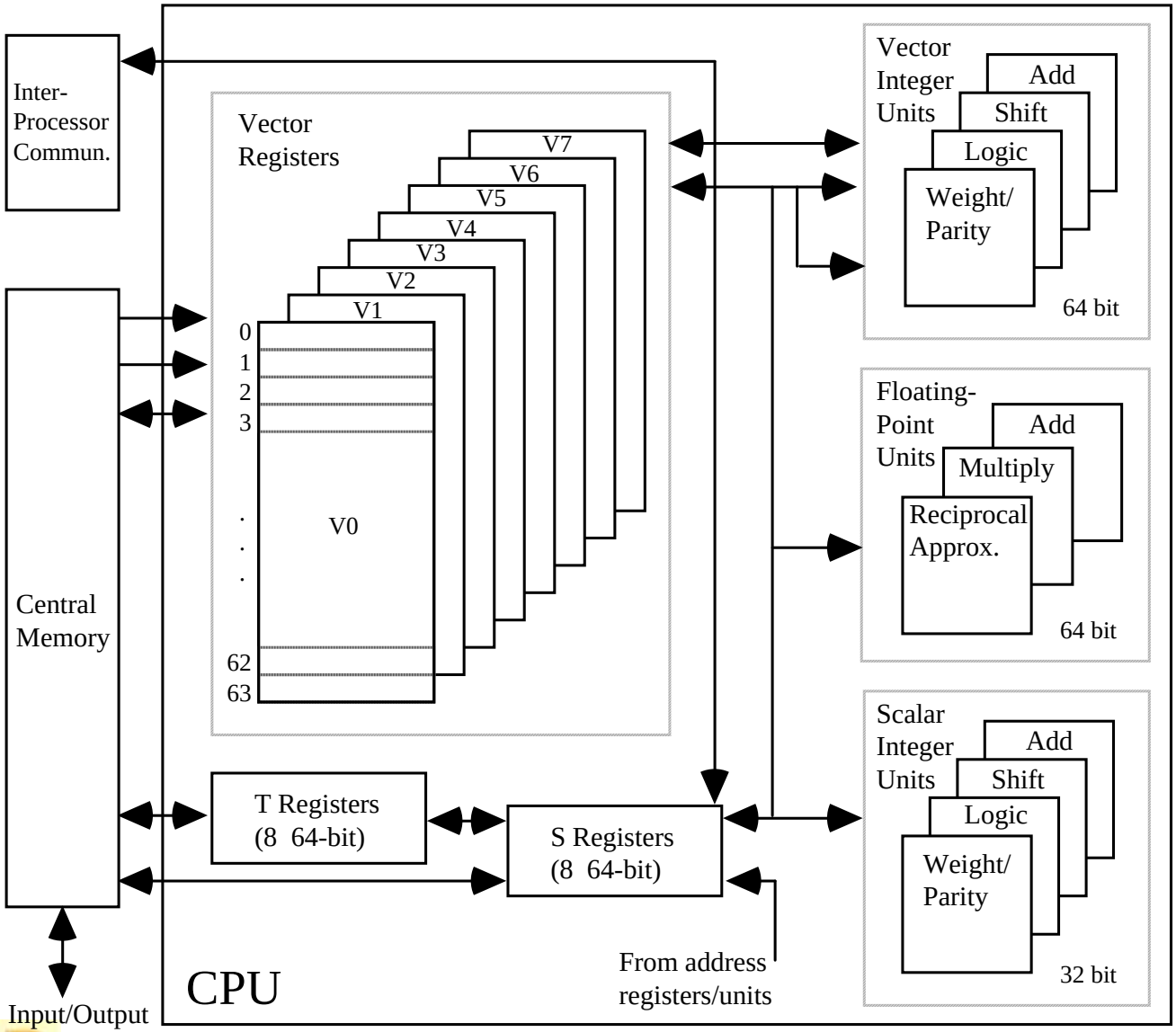



Fig. 21.4 A small 16-node version of the multistage interconnection network of the BBN Butterfly.

# 21.3 Vector-Parallel Cray Y-MP

Fig. 21.5 Key elements of the Cray Y-MP processor. Address registers, address function units, instruction buffers, and control not shown.



# Cray Y-MP's Interconnection Network

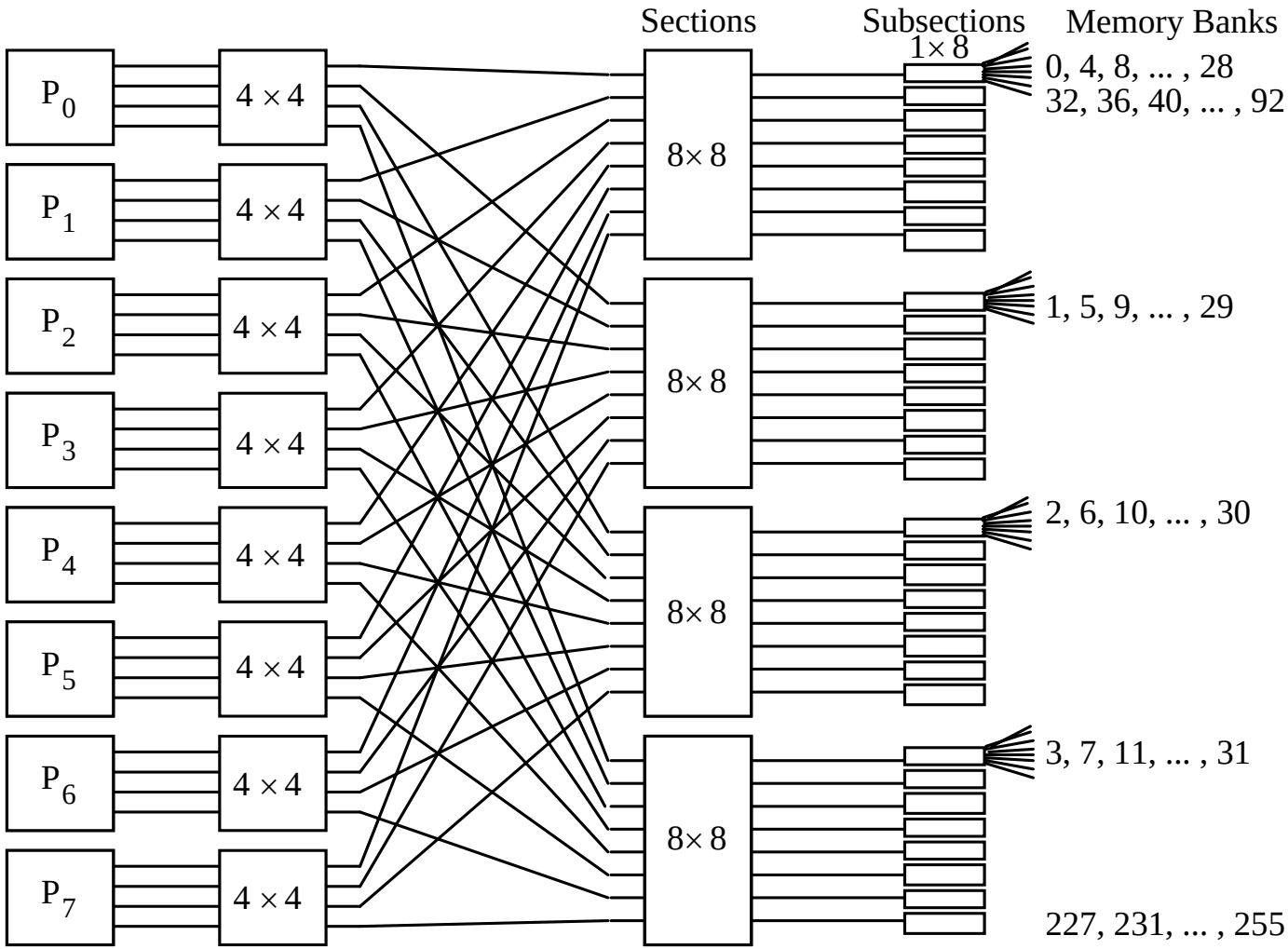



Fig. 21.6 The processor-to-memory interconnection network of Cray Y-MP.

# 21.4 Latency-Tolerant Tera MTA

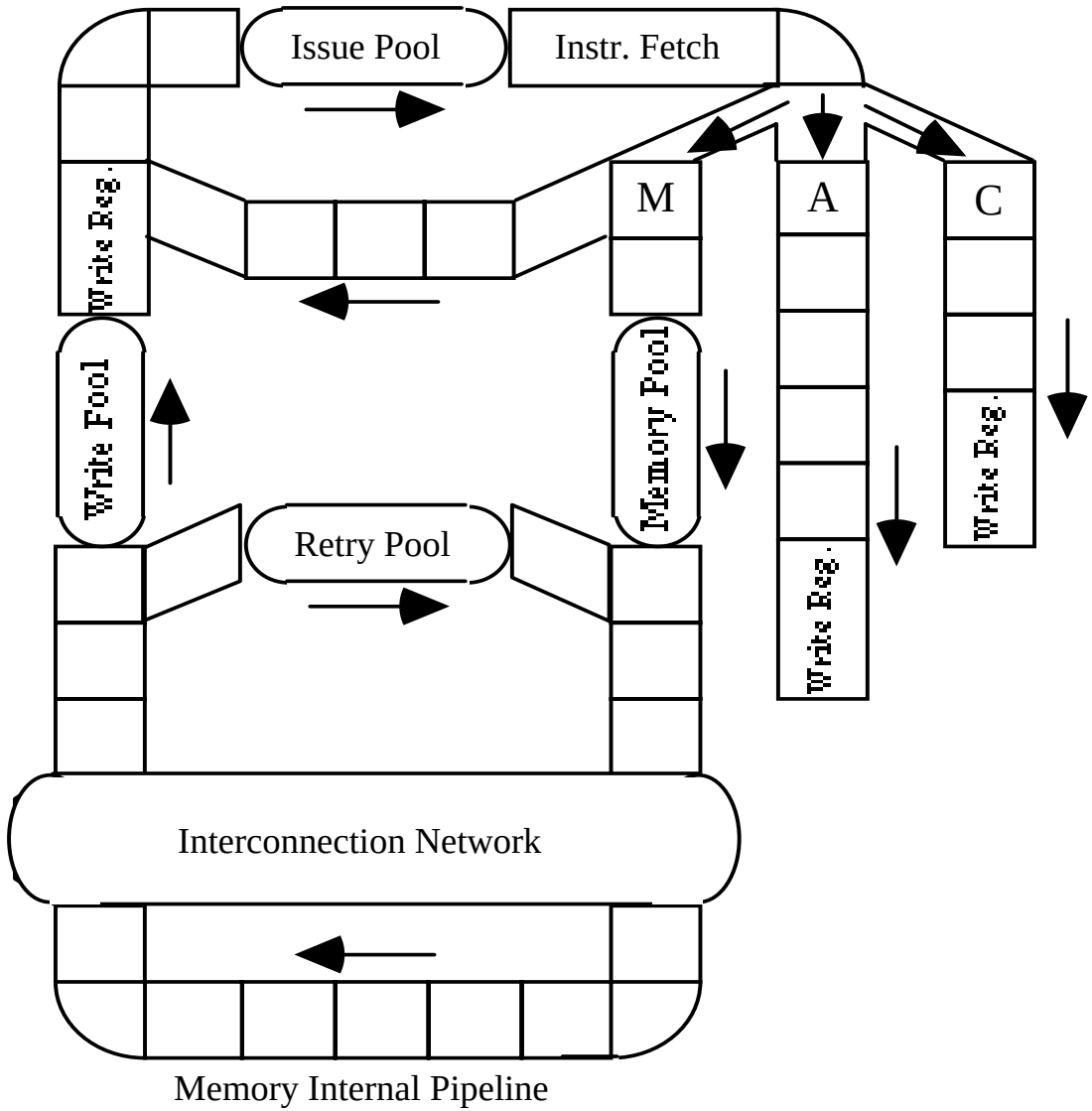



Fig. 21.7
The instruction execution pipelines of Tera MTA.

# 21.5 CC-NUMA Stanford DASH

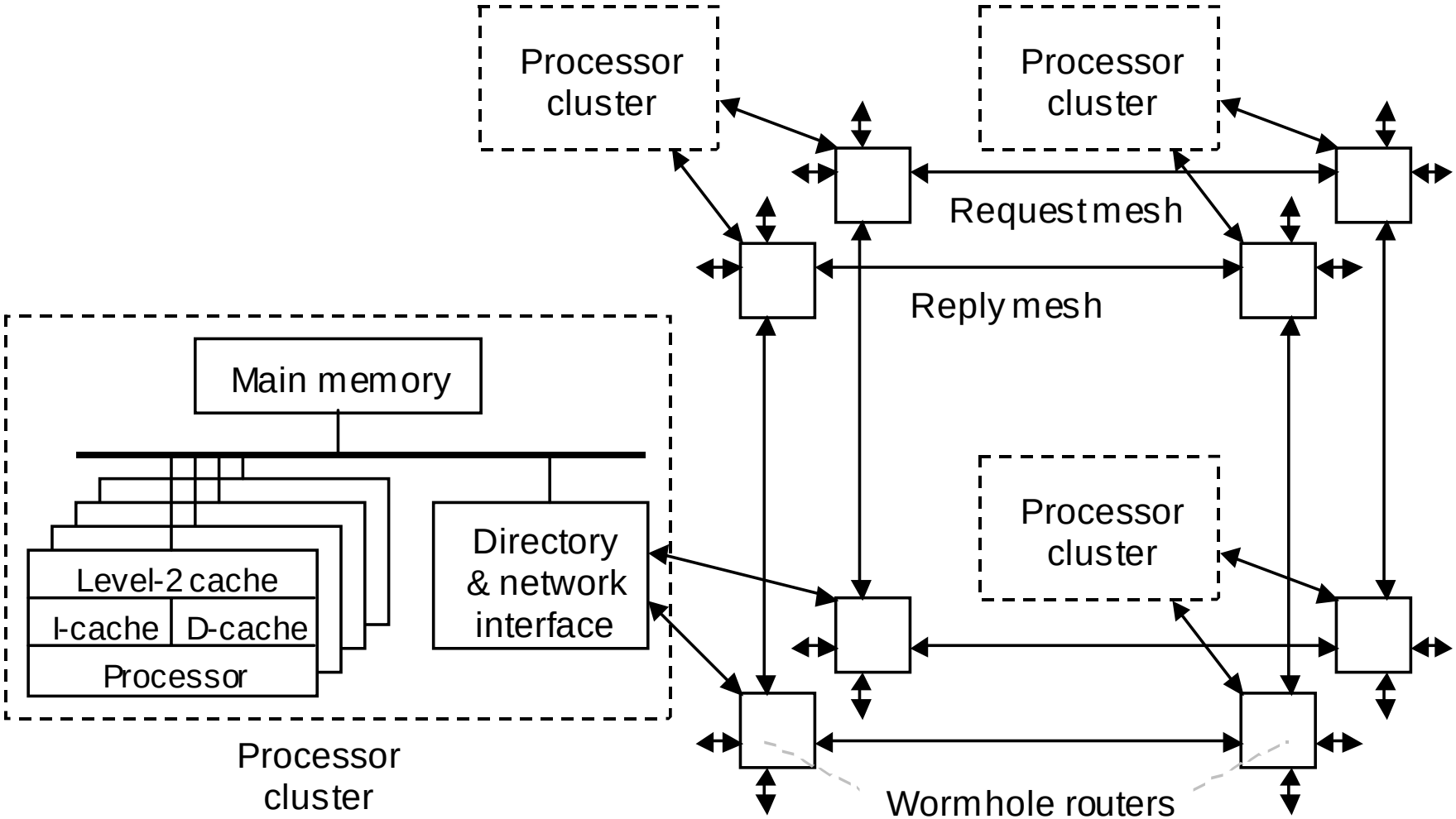
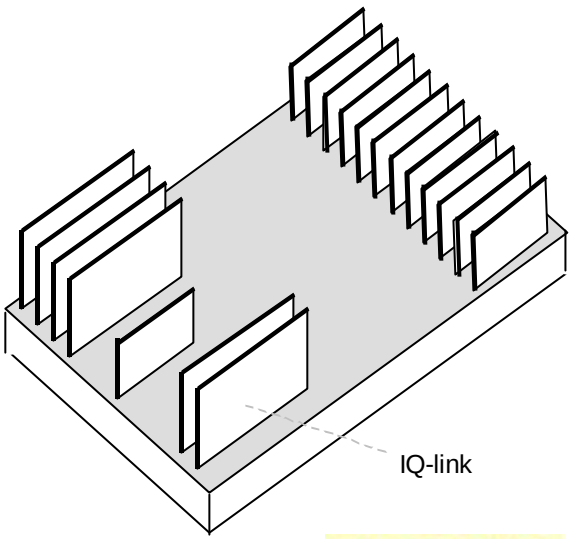



Fig. 21.8 The architecture of Stanford DASH.

# 21.6 SCI-Based Sequent NUMA-Q

Fig. 21.9 The physical placement of Sequent's quad components on a rackmount baseboard (not to scale)



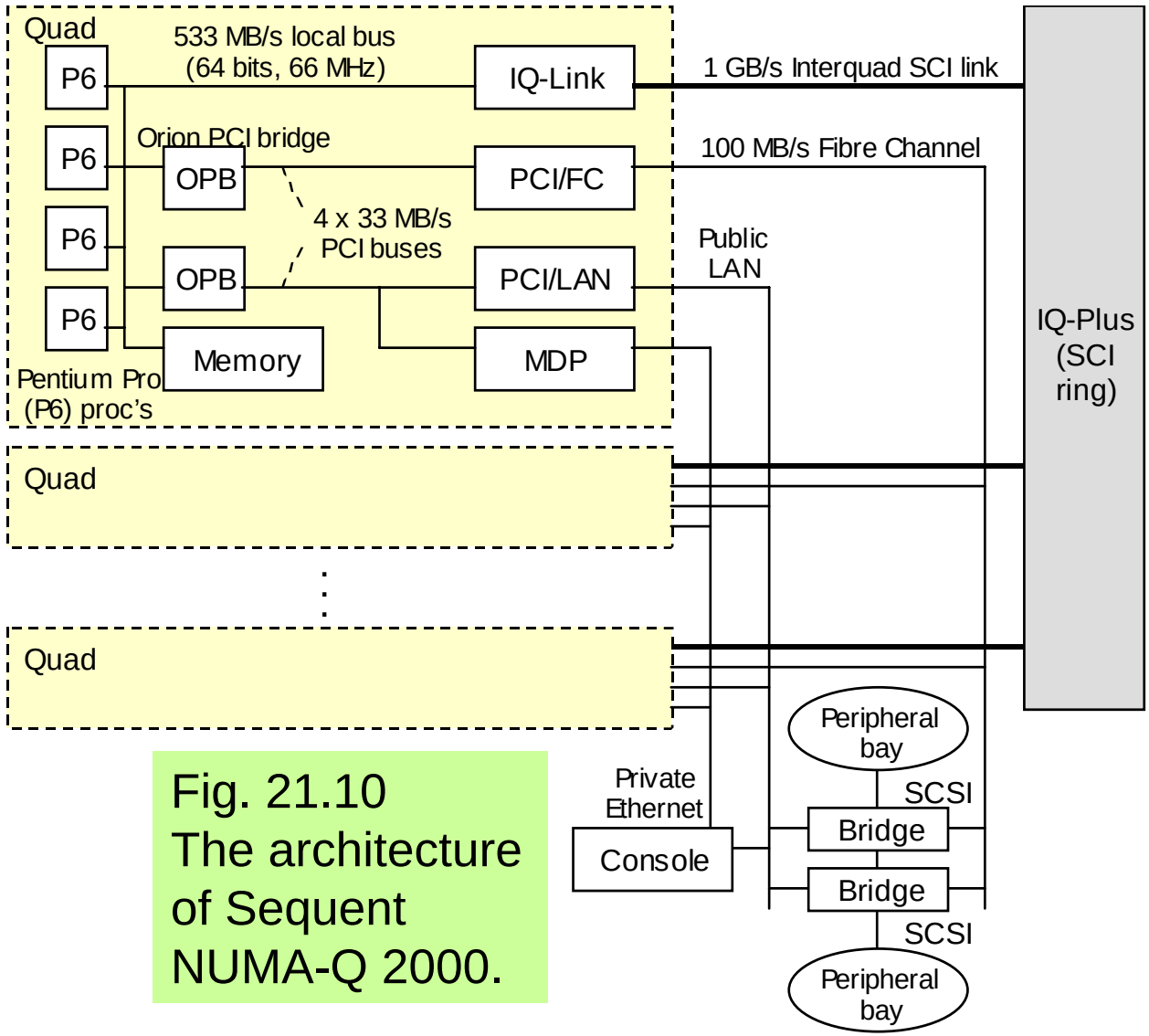




Fig. 21.10 The architecture of Sequent NUMA-Q 2000.

# Details of the IQ-Link in Sequent NUMA-Q

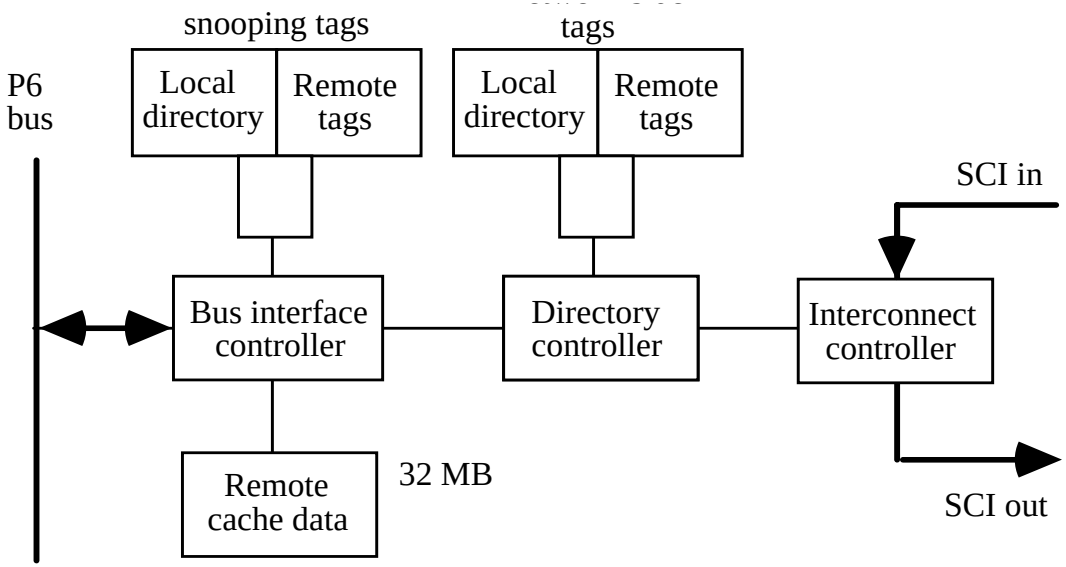



Fig. 21.11
Block diagram of the IQ-Link board.

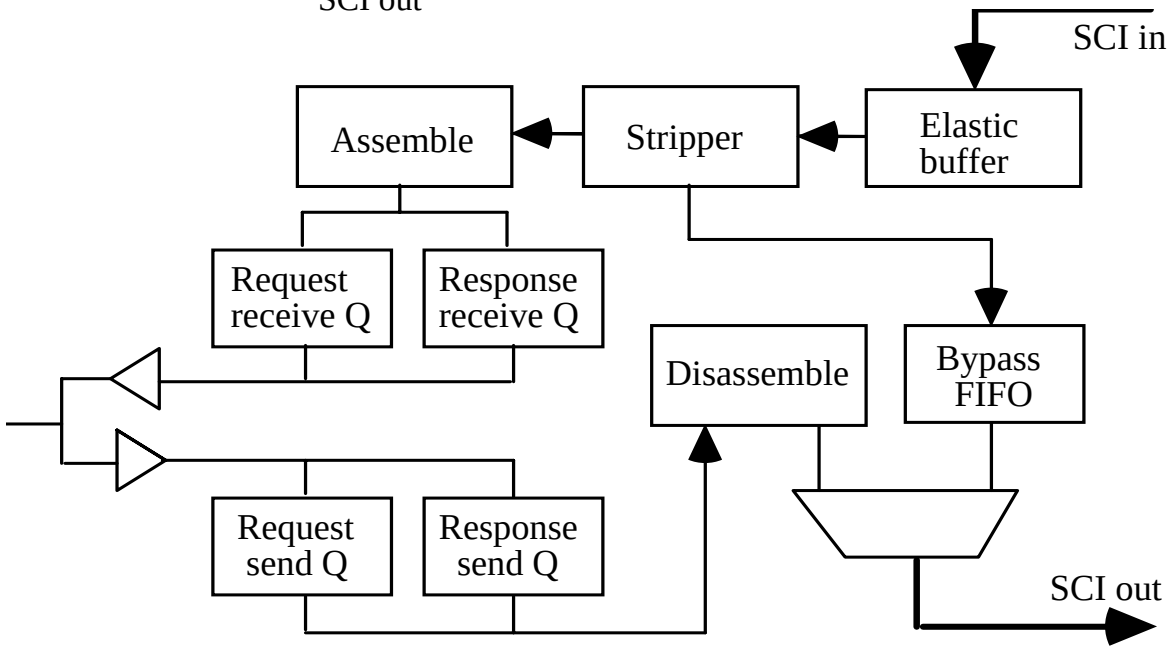



Fig. 21.12
Block diagram of IQ-Link's interconnect controller.

# 22 Message-Passing MIMD Machines

Learn about practical message-passing parallel architectures:

- Study mechanisms that support message passing
- Examine research prototypes and production machines

| Topics in This Chapter | |
|---|---|
| 22.1 | Mechanisms for Message Passing |
| 22.2 | Reliable Bus-Based Tandem NonStop |
| 22.3 | Hypercube-Based nCUBE3 |
| 22.4 | Fat-Tree-Based Connection Machine 5 |
| 22.5 | Omega-Network-Based IBM SP2 |
| 22.6 | Commodity-Driven Berkeley NOW |

# 22.1 Mechanisms for Message Passing

1. **Shared-Medium networks** (buses, LANs; e.g., Ethernet, token-based)
2. **Router-based networks** (aka direct networks; e.g., Cray T3E, Chaos)
3. **Switch-based networks** (crossbars, MINs; e.g., Myrinet, IBM SP2)

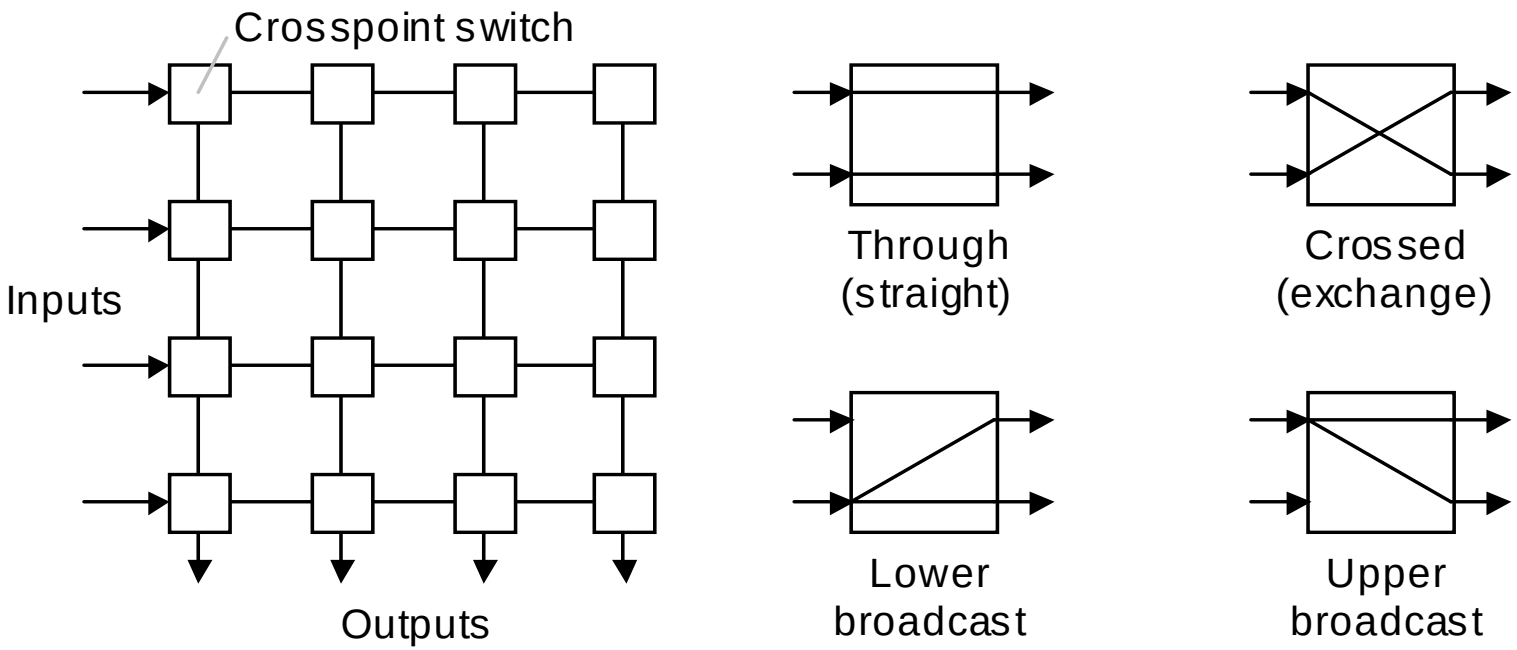



Fig. 22.2 Example 4 × 4 and 2 × 2 switches used as building blocks for larger networks.

# Router-Based Networks

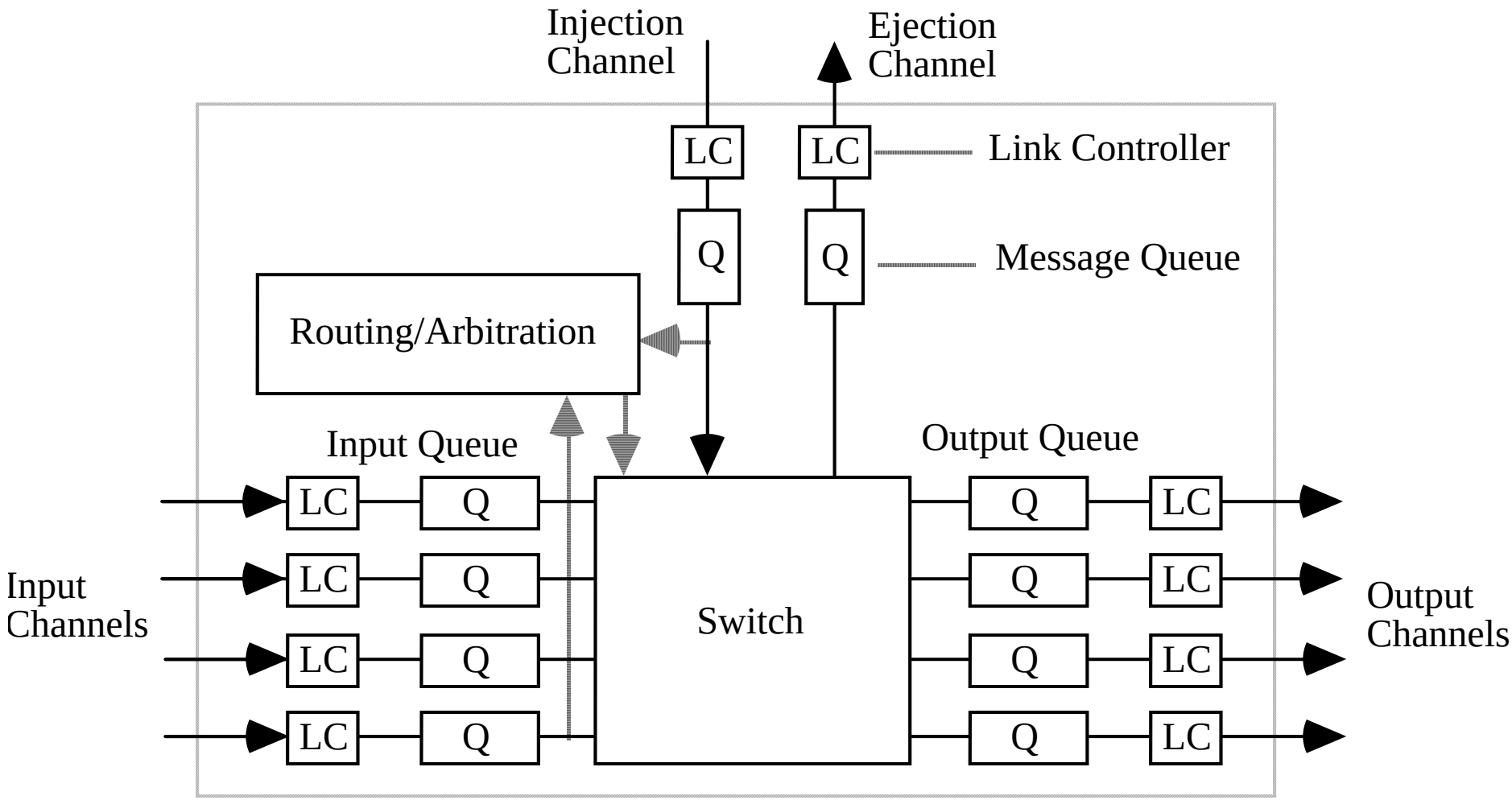



Fig. 22.1 The structure of a generic router.

# Case Studies of Message-Passing Machines

Fig. 22.3 Classification of message-passing hardware architectures and example systems that will be studied in this chapter.

| | Shared-Medium Network | Router-Based Network | Switch-Based Network |
|---|---|---|---|
| Coarse-Grain | Tandem NonStop (Bus) | | |
| Medium-Grain | Berkeley NOW (LAN) | nCUBE3 | TMC CM-5<br>IBM SP2 |
| Fine-Grain | | | |

# 22.2 Reliable Bus-Based Tandem NonStop

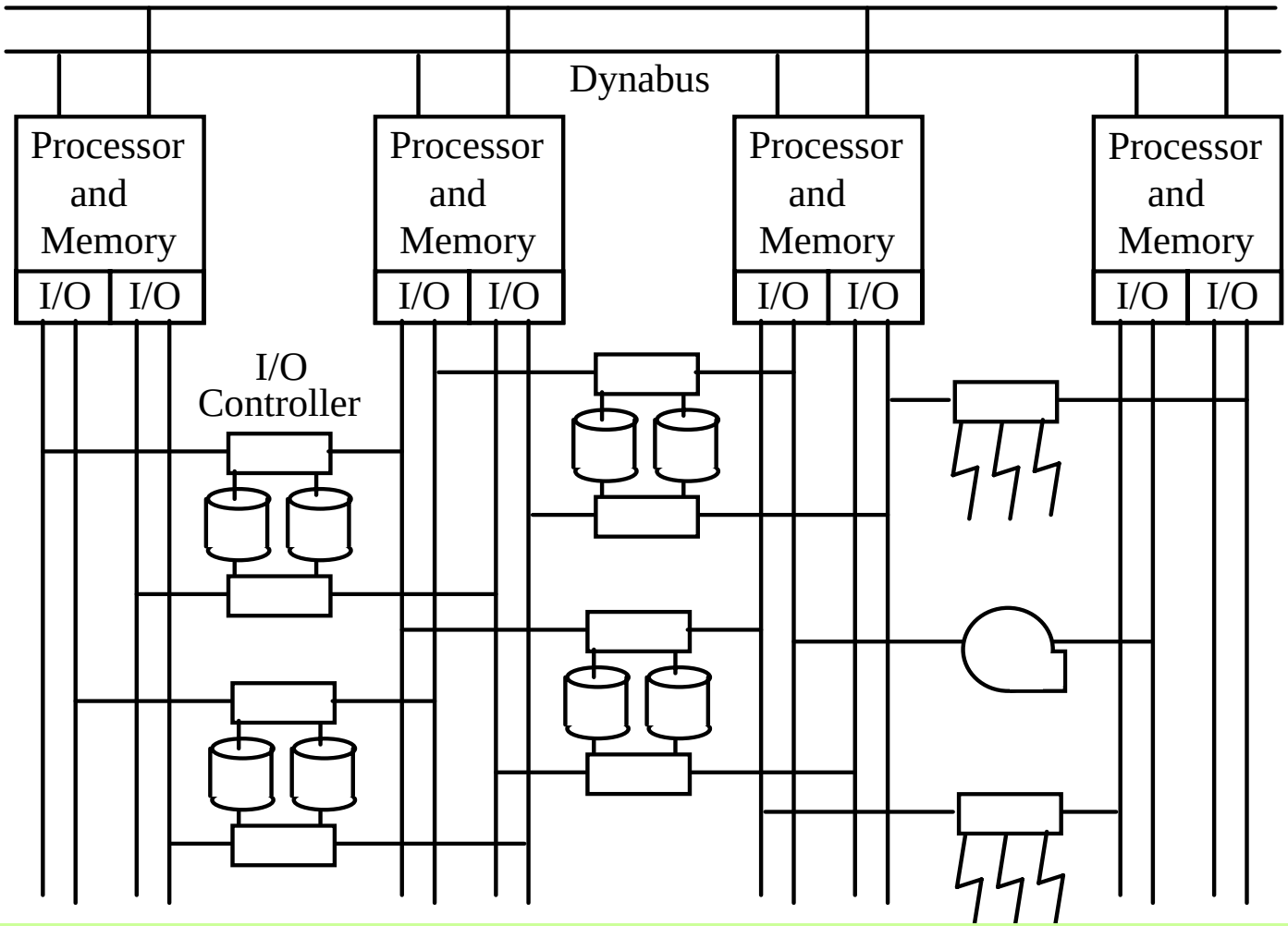



Fig. 22.4 One section of the Tandem NonStop Cyclone system.

# High-Level Structure of the Tandem NonStop

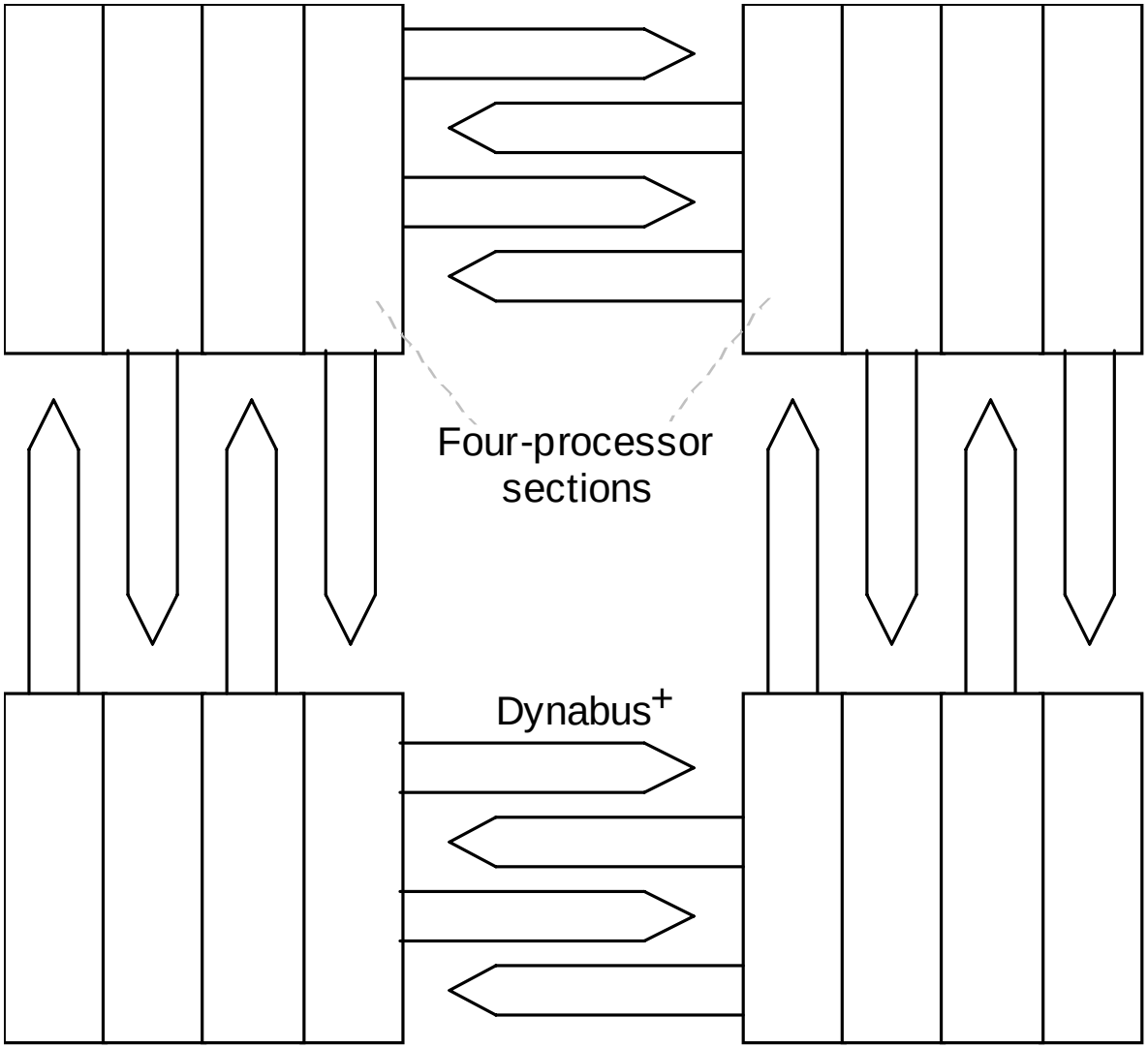



Fig. 22.5 Four four-processor sections interconnected by Dynabus+.

# 22.3 Hypercube-Based nCUBE3

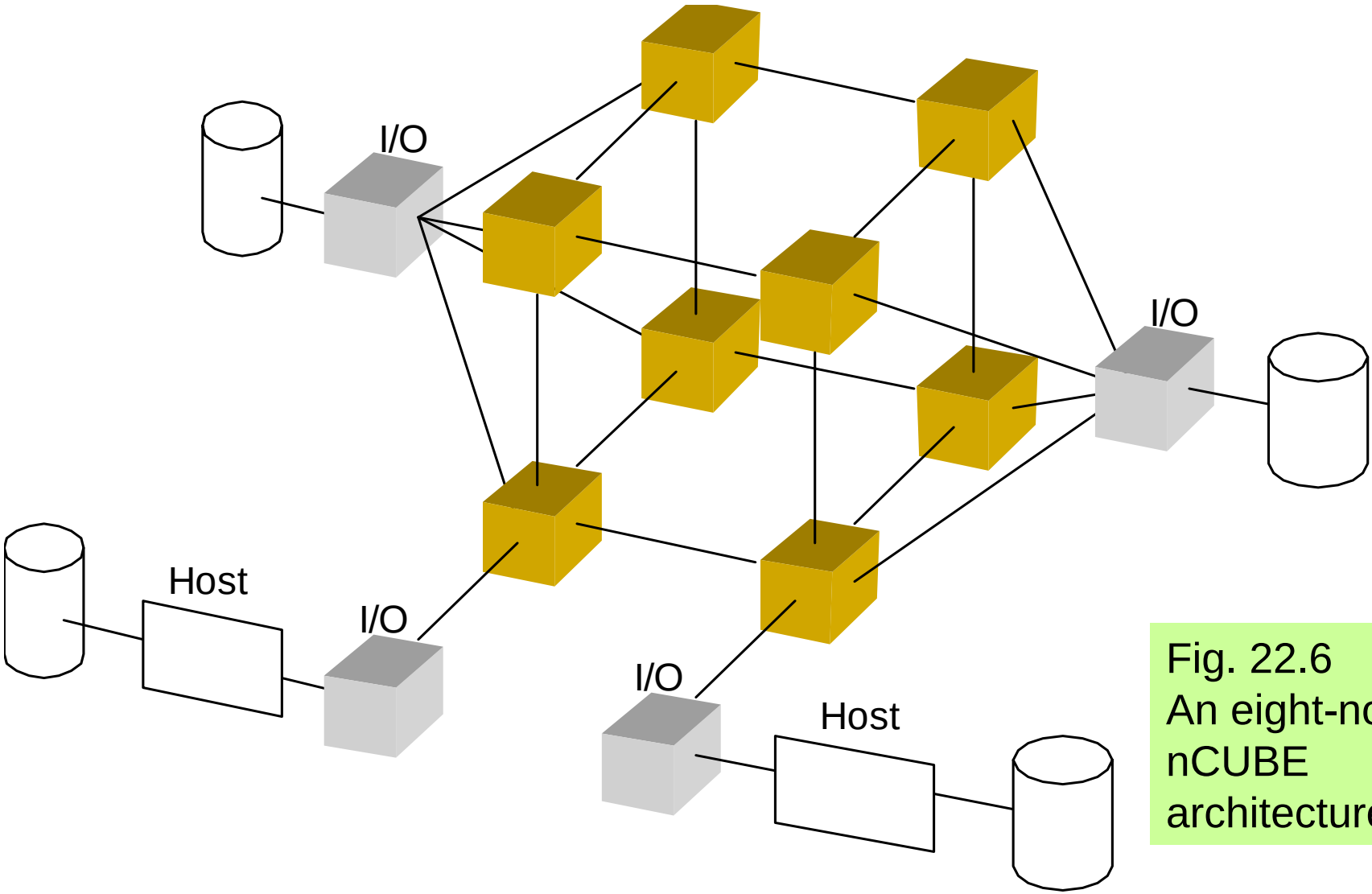



Fig. 22.6
An eight-node nCUBE architecture.

# 22.4 Fat-Tree-Based Connection Machine 5

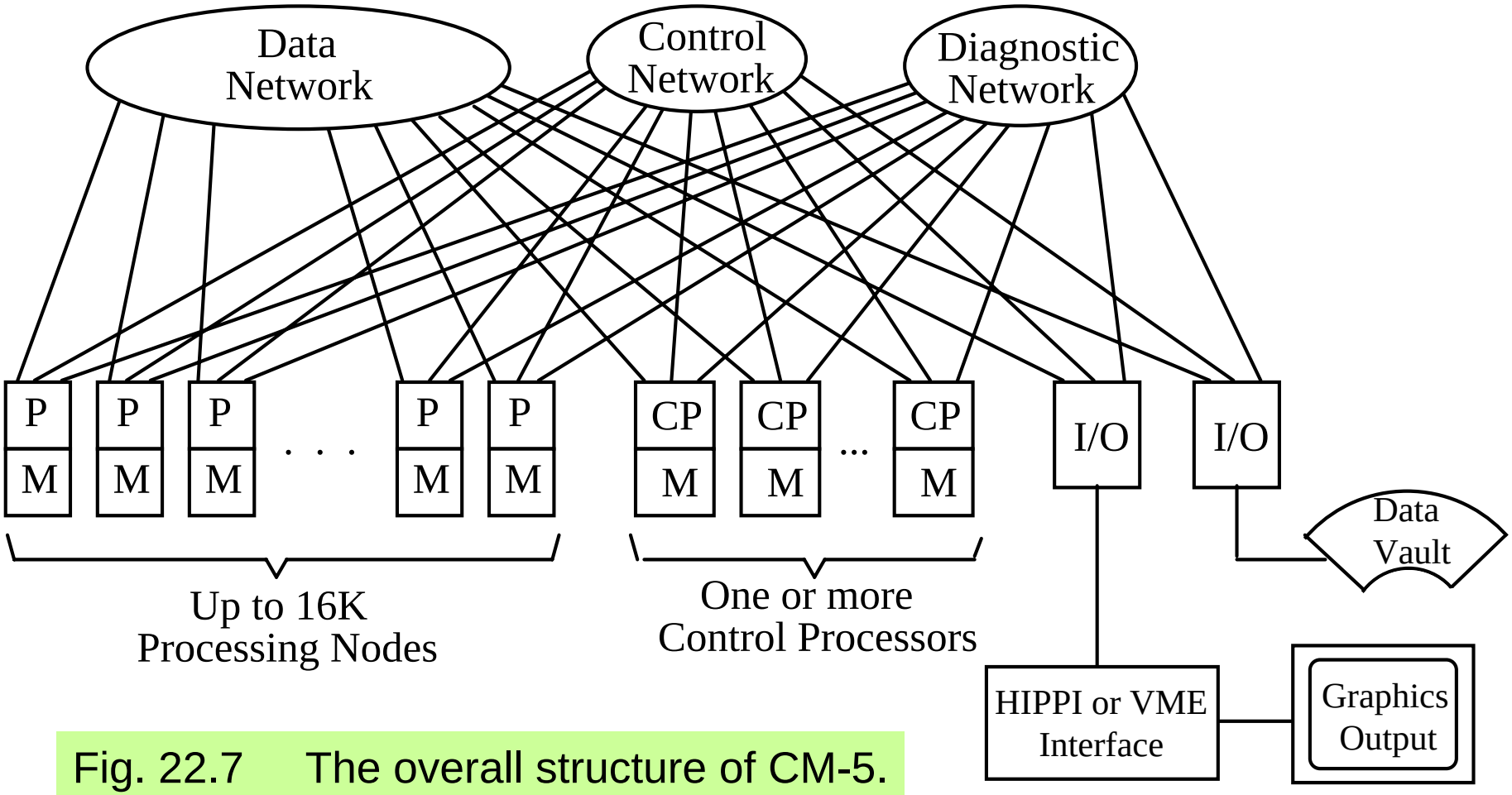



Fig. 22.7 The overall structure of CM-5.

# Processing Nodes in CM-5

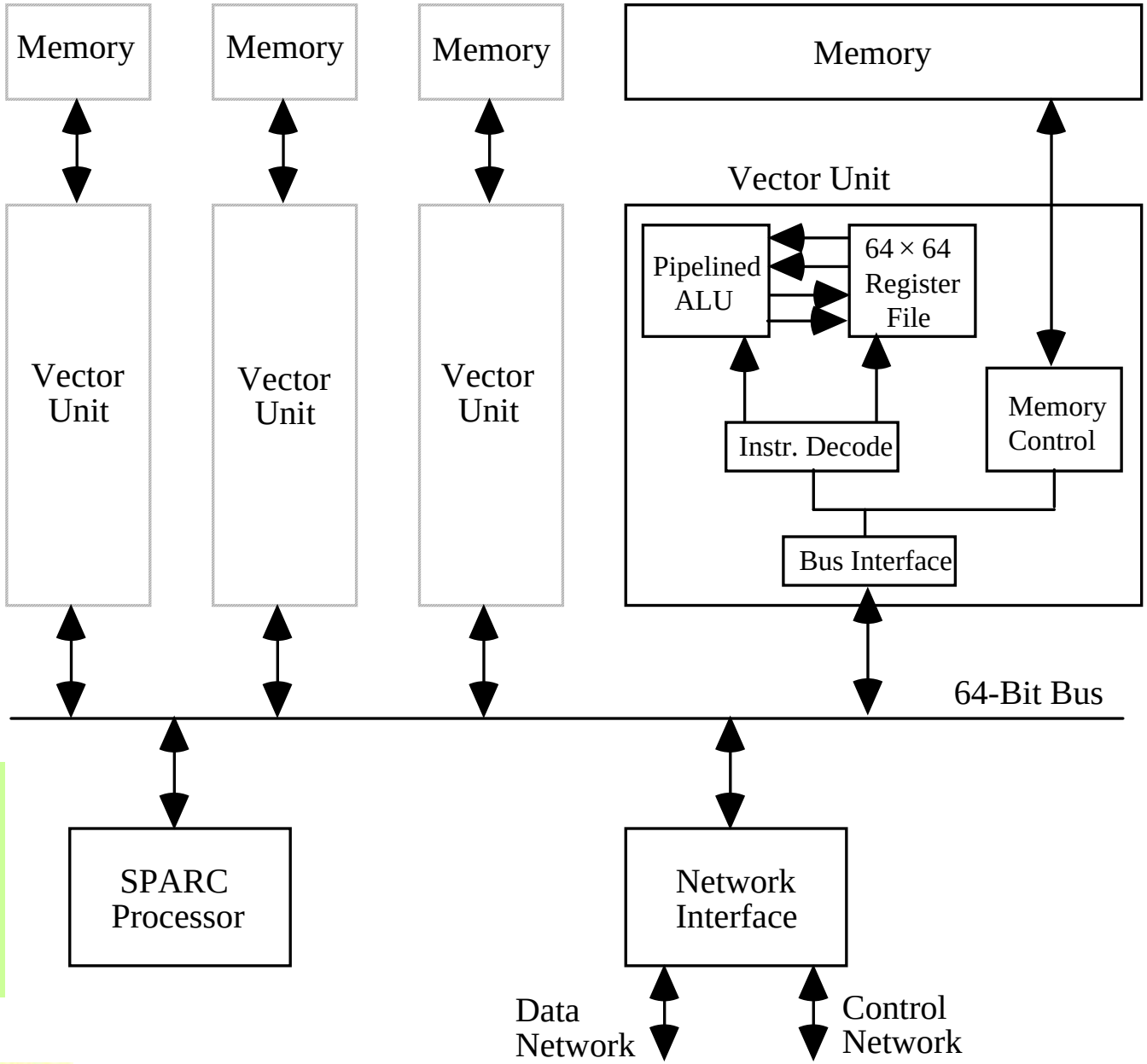



Fig. 22.8
The components of a processing node in CM-5.

# The Interconnection Network in CM-5

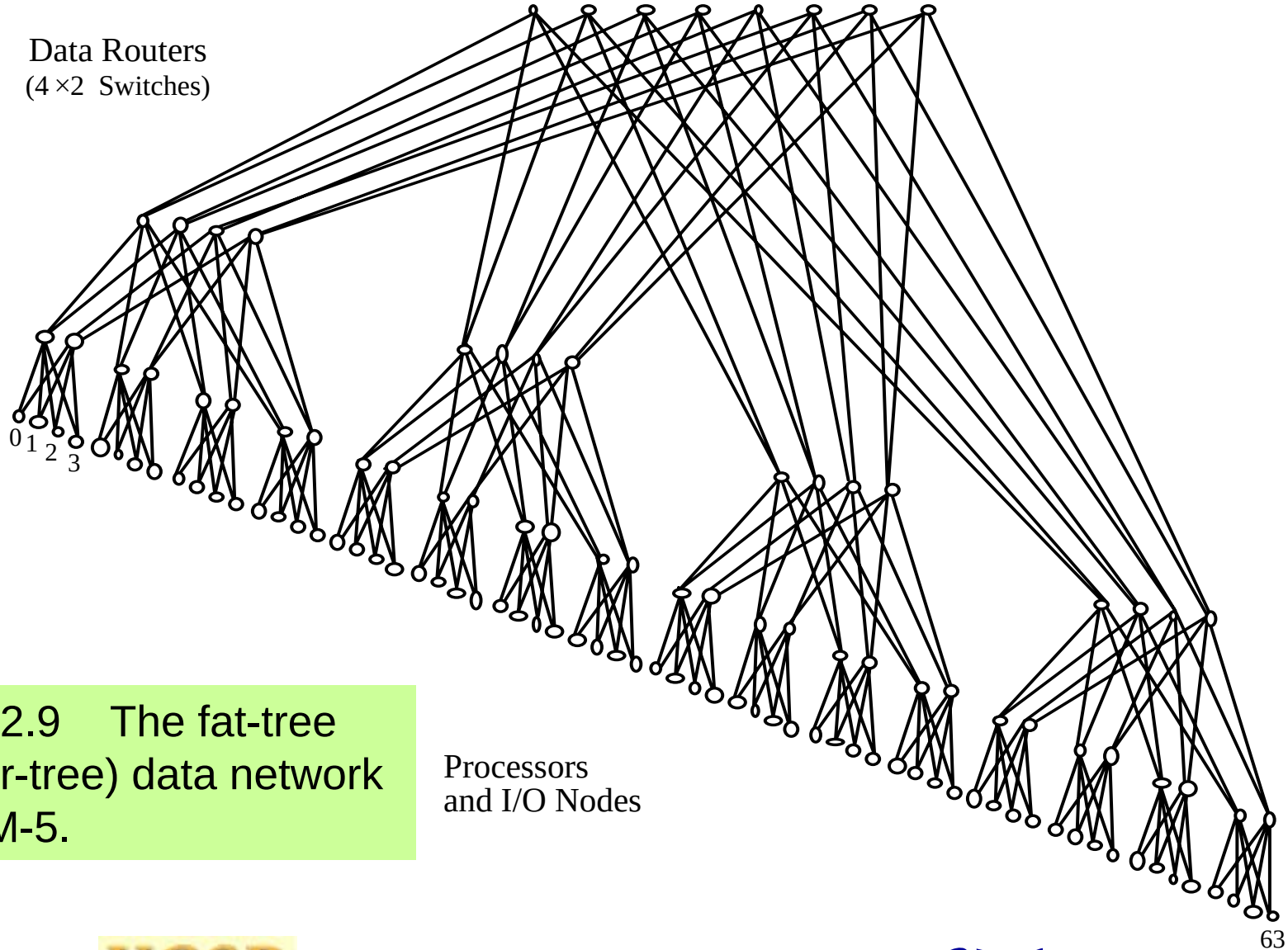



Fig. 22.9 The fat-tree (hyper-tree) data network of CM-5.

# 22.5 Omega-Network-Based IBM SP2

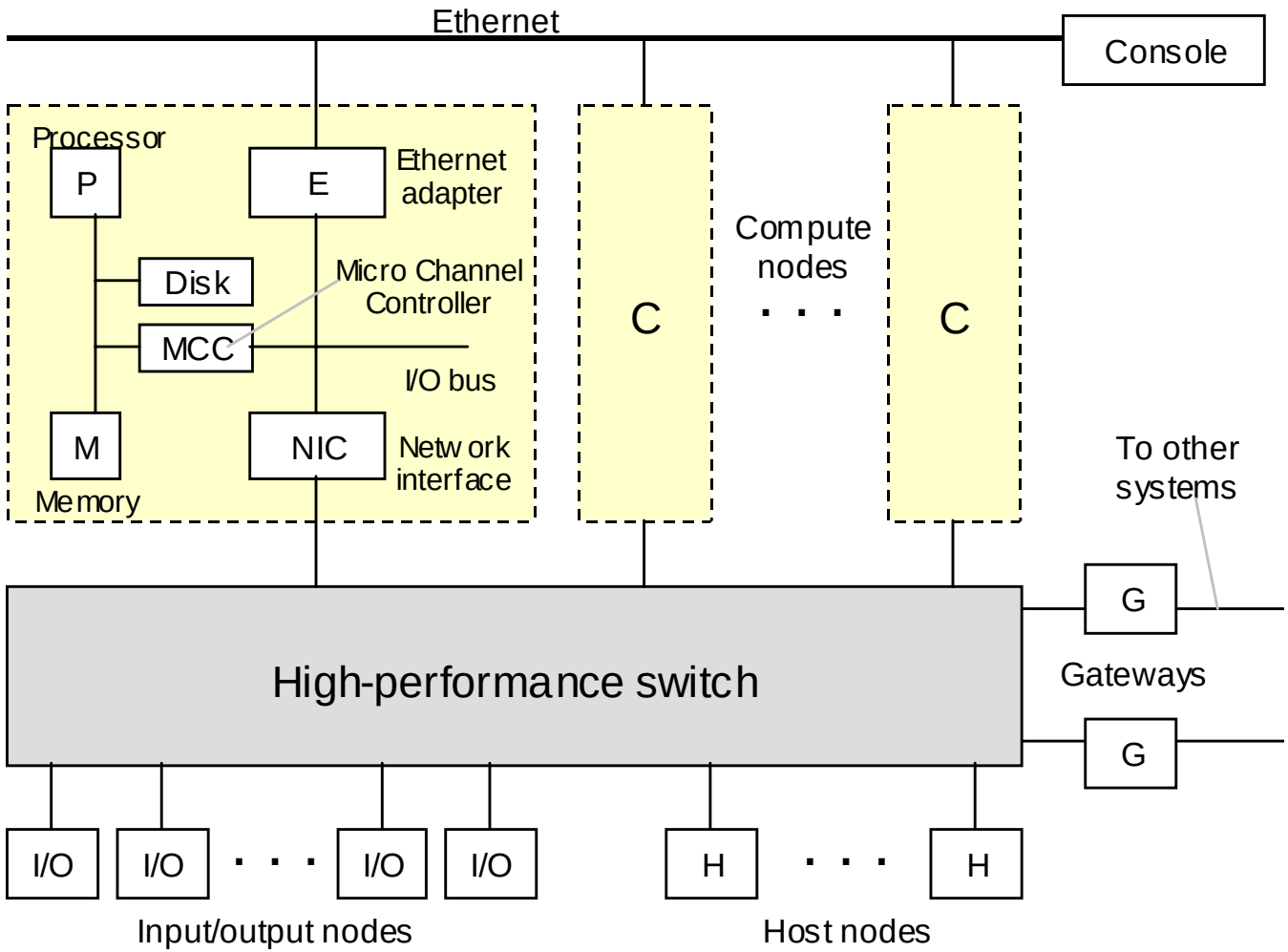



Fig. 22.10 The architecture of IBM SP series of systems.

# The IBM SP2 Network Interface

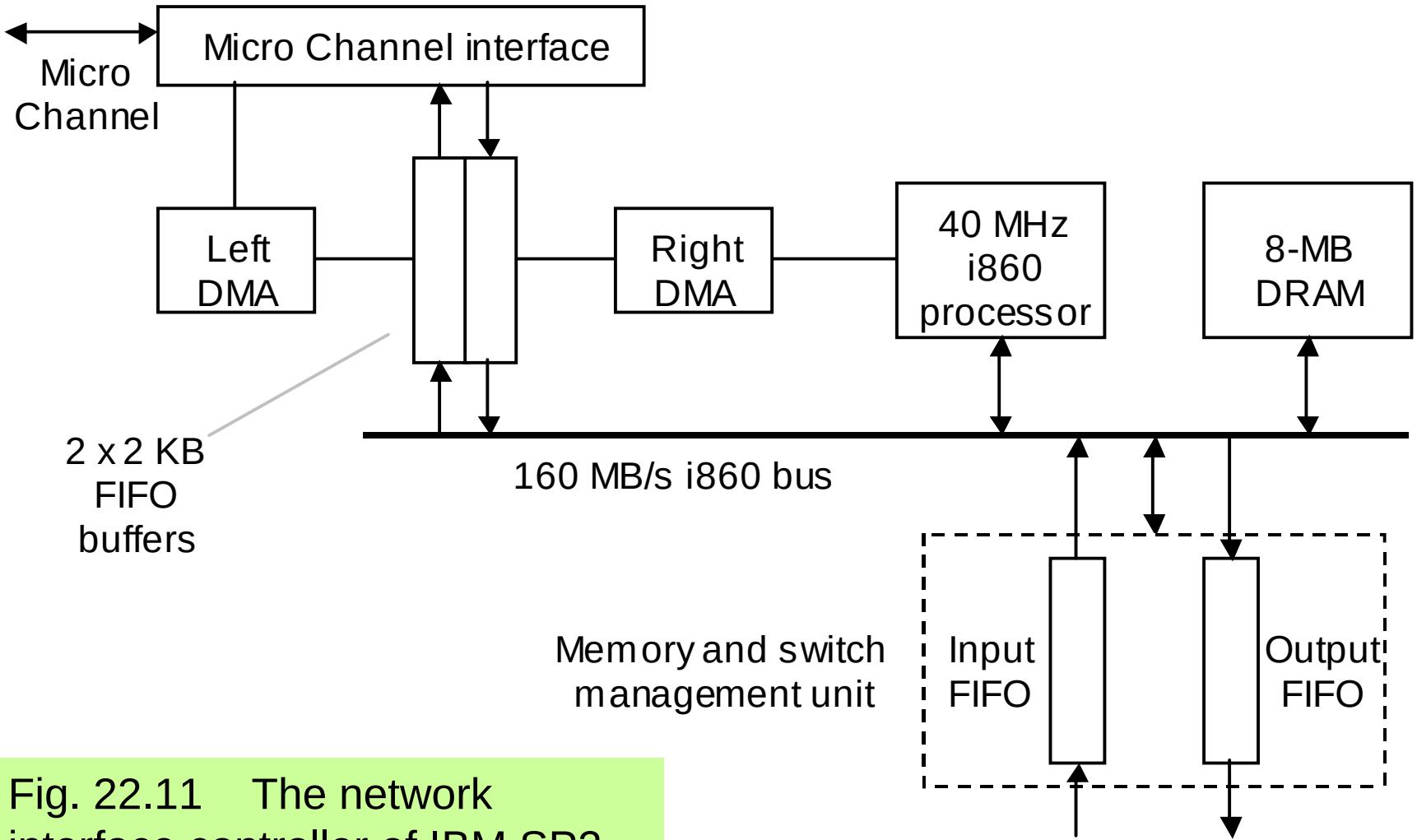



Fig. 22.11 The network interface controller of IBM SP2.

# The Interconnection Network of IBM SP2

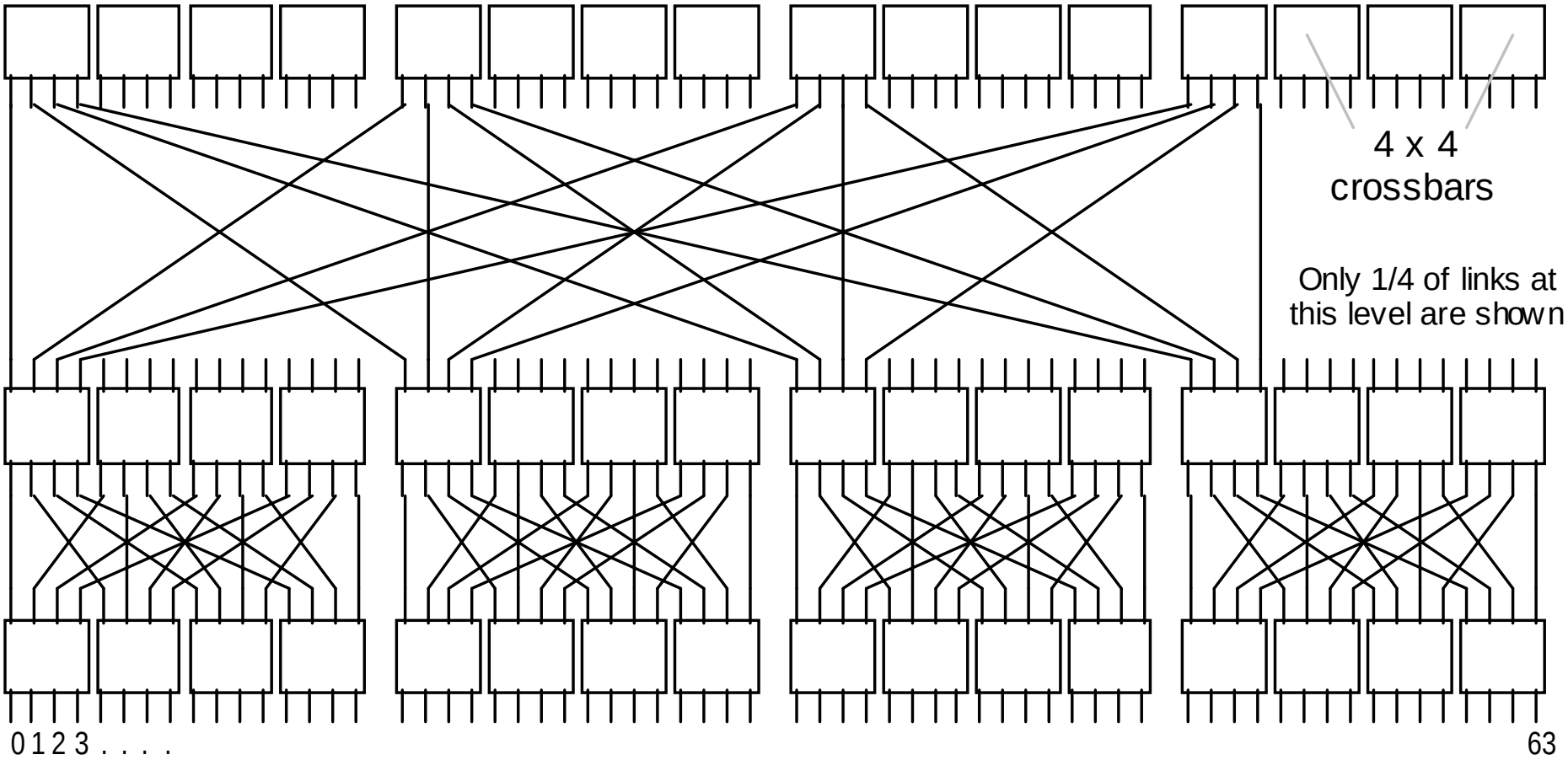



Fig. 22.12 A section of the high-performance switch network of IBM SP2.

# 22.6 Commodity-Driven Berkeley NOW

NOW aims for building a distributed supercomputer from commercial workstations (high end PCs), linked via switch-based networks

**Components and challenges for successful deployment of NOWs (aka clusters of workstations, or COWs)**

Network interface hardware: System-area network (SAN), middle ground between LANs and specialized interconnections

Fast communication protocols: Must provide competitive parameters for the parameters in the LogP model (see Section 4.5)

Distributed file systems: Data files must be distributed among nodes, with no central warehouse or arbitration authority

Global resource management: Berkeley uses GLUnix (global-layer Unix); other systems of comparable functionalities include Argonne Globus, NSCP metacomputing system, Princeton SHRIMP, Rice TreadMarks, Syracuse WWVM, Virginia Legion, Wisconsin Wind Tunnel

# 23 Data-Parallel SIMD Machines

Learn about practical implementation of SIMD architectures:

- Assess the successes, failures, and potentials of SIMD
- Examine various SIMD parallel computers, old and new

| Topics in This Chapter | |
|---|---|
| 23.1 | Where Have All the SIMDs Gone? |
| 23.2 | The First Supercomputer: ILLIAC IV |
| 23.3 | Massively Parallel Goodyear MPP |
| 23.4 | Distributed Array Processor (DAP) |
| 23.5 | Hypercubic Connection Machine 2 |
| 23.6 | Multiconnected MasPar MP-2 |

# 23.1 Where Have All the SIMDs Gone?

Associative or content-addressable memories/processors constituted early forms of SIMD parallel processing

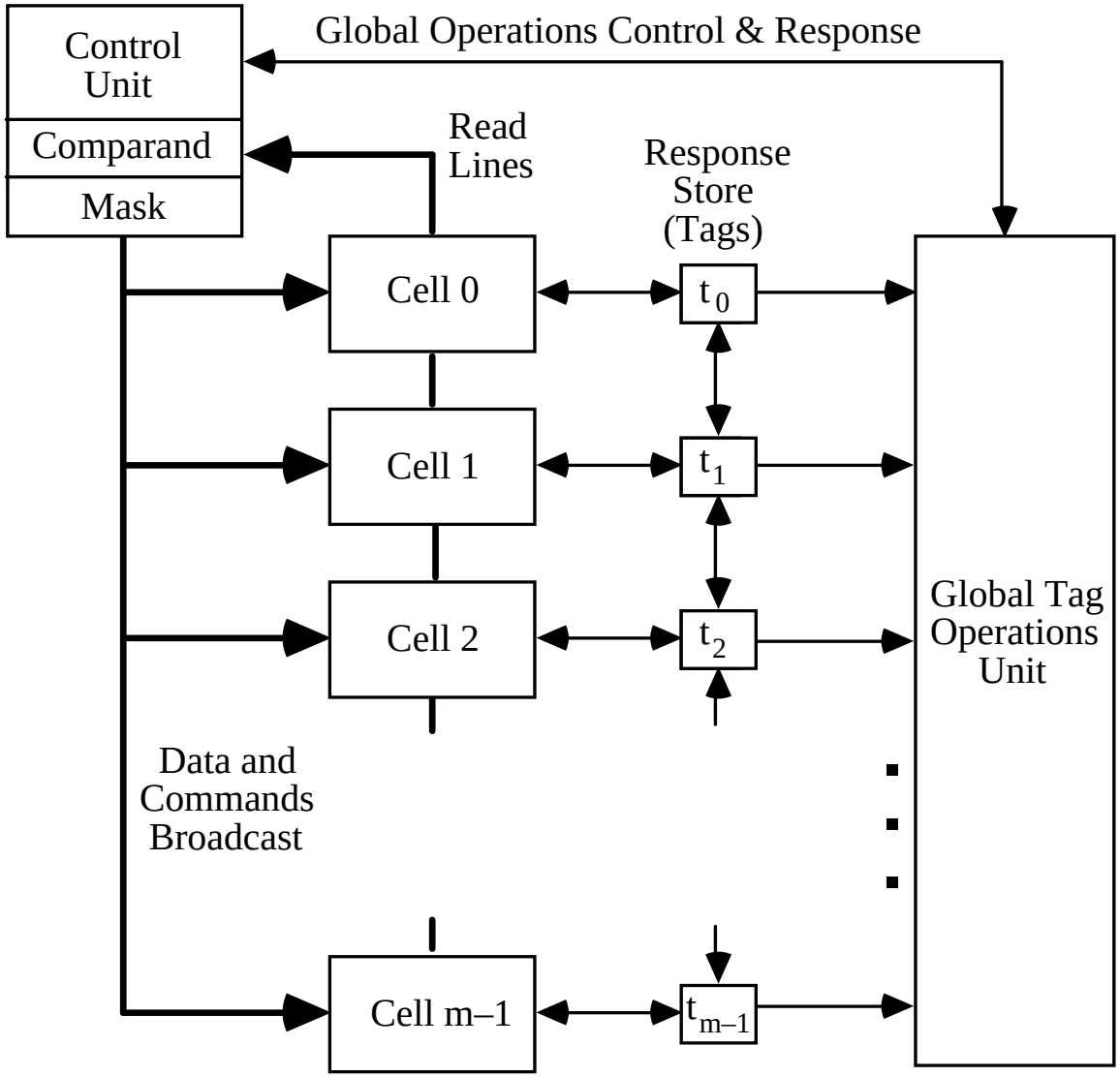



Fig. 23.1
Functional view of an associative memory/processor.

# Search Functions in Associative Devices

*Exact match:* Locating data based on partial knowledge of contents

*Inexact match:* Finding numerically or logically proximate values

*Membership:* Identifying all members of a specified set

*Relational:* Determining values that are less than, less than or equal, etc.

*Interval:* Marking items that are between or outside given limits

*Extrema:* Finding the maximum, minimum, next higher, or next lower

*Rank-based:* Selecting *k*th or *k* largest/smallest elements

*Ordered retrieval:* Repeated max- or min-finding with elimination (sorting)

# Mixed-Mode SIMD/MIMD Parallelism

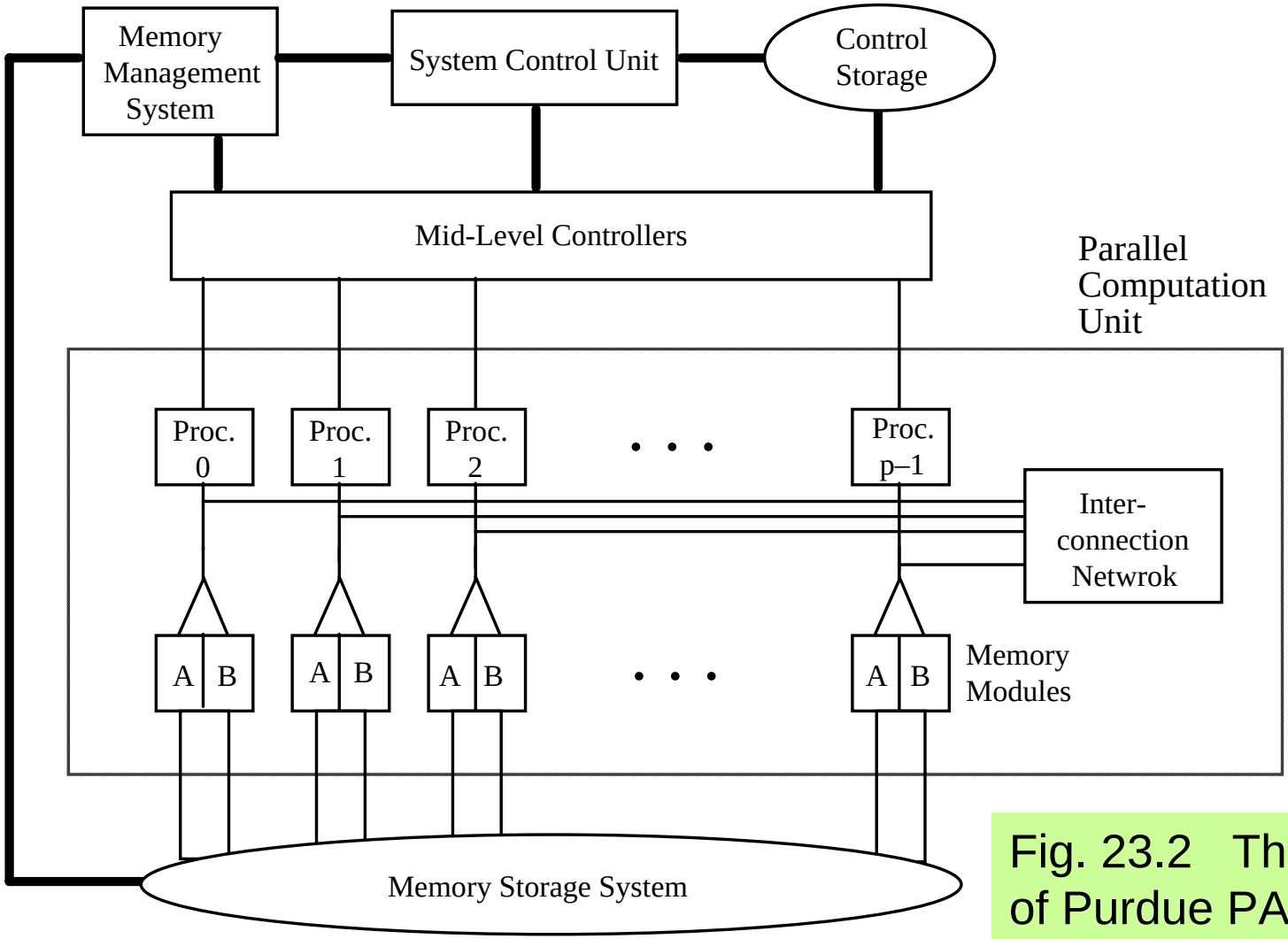



Fig. 23.2 The architecture of Purdue PASM.

# 23.2 The First Supercomputer: ILLIAC IV

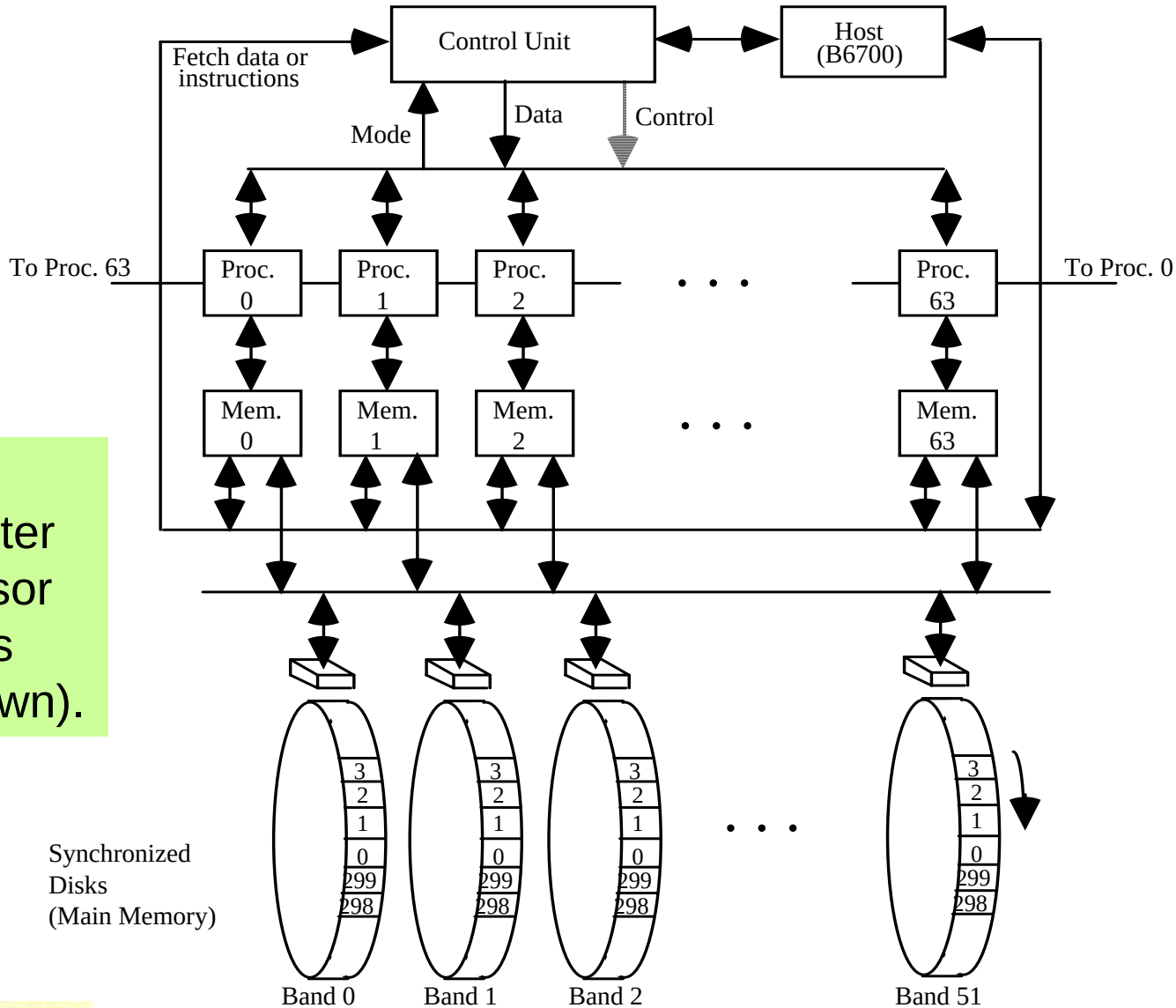



Fig. 23.3 The ILLIAC IV computer (the inter-processor routing network is only partially shown).

# 23.3 Massively Parallel Goodyear MPP

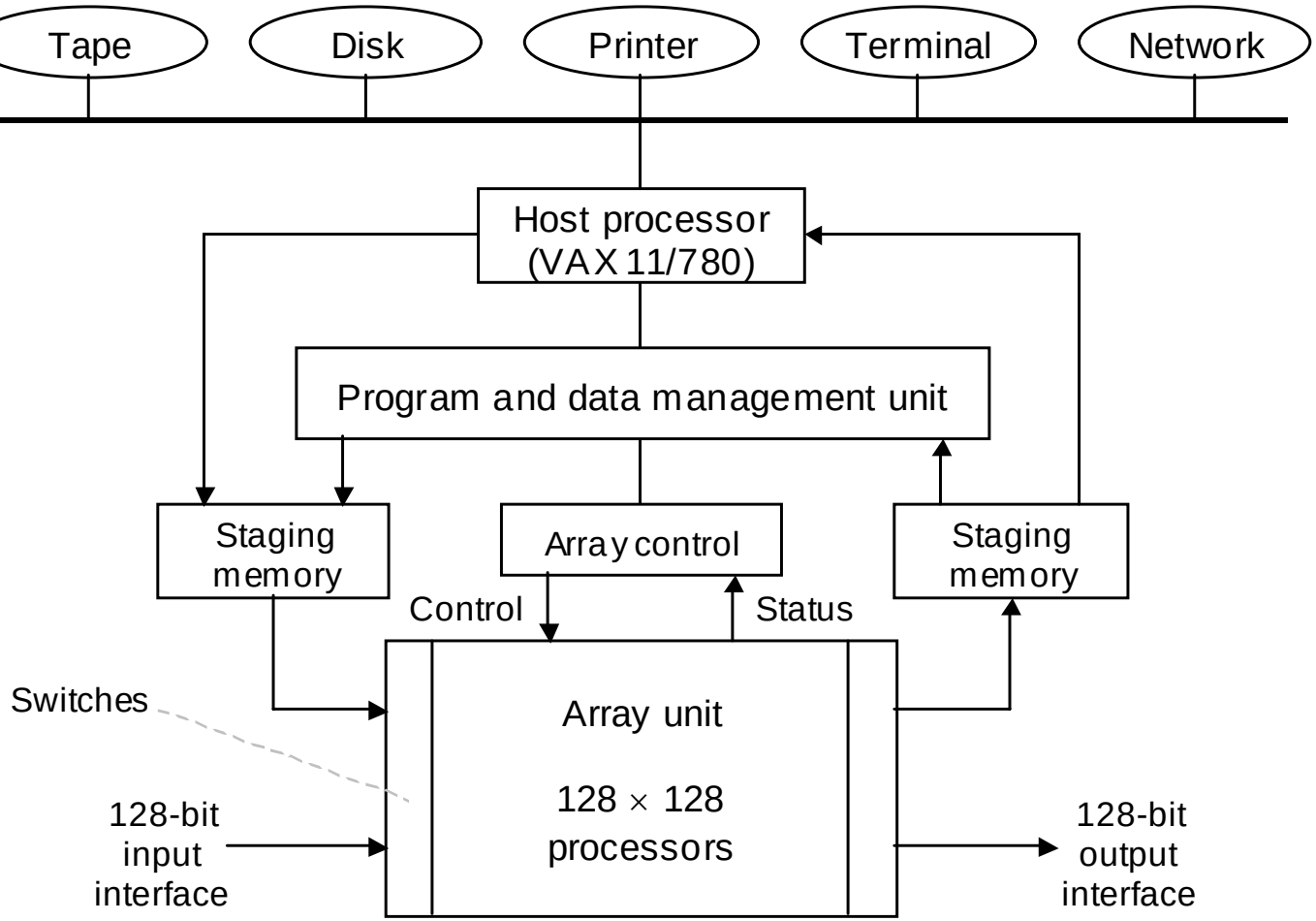



Fig. 23.4 The architecture of Goodyear MPP.

# Processing Elements in the Goodyear MPP

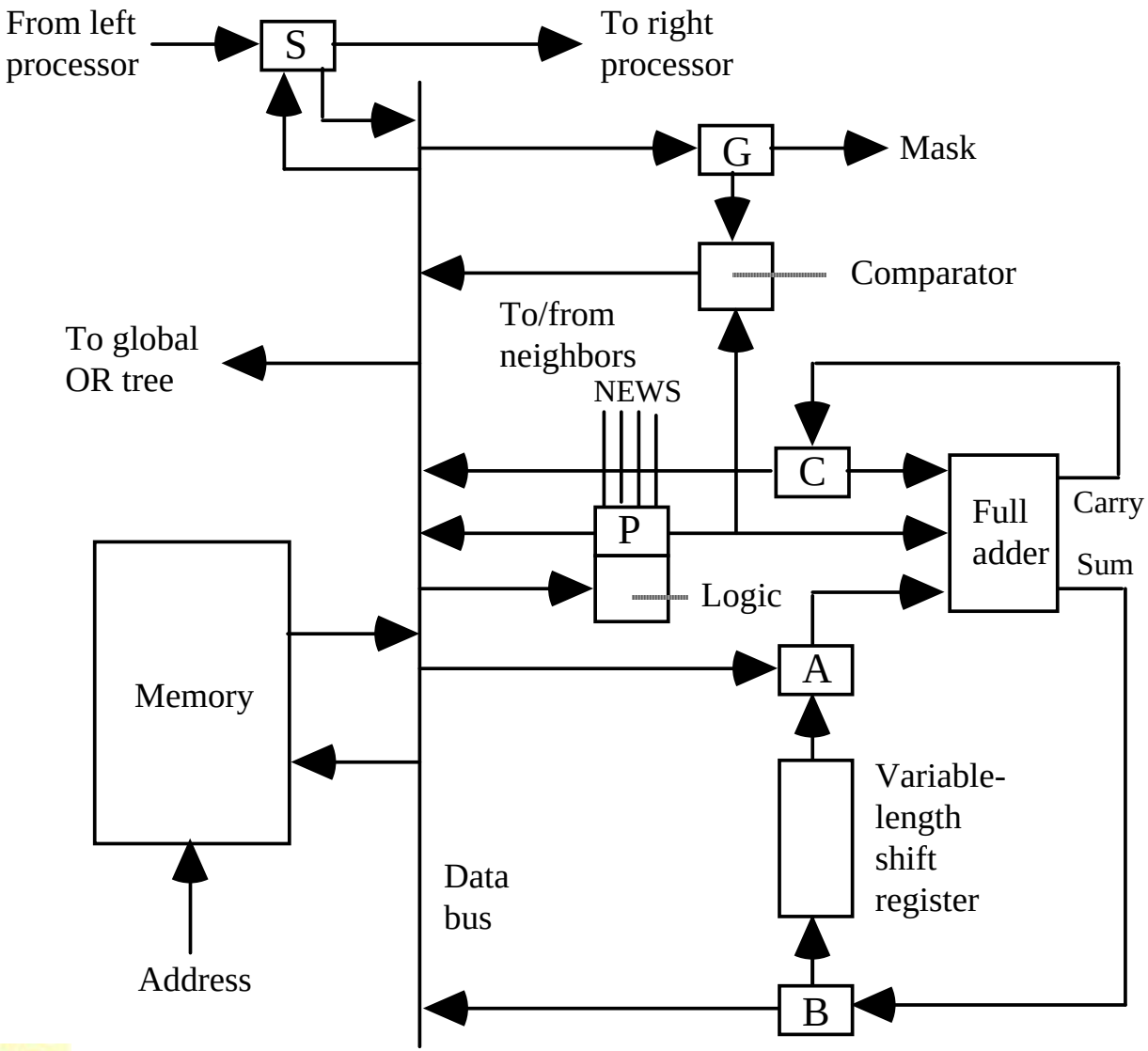



Fig. 23.5
The single-bit processor of MPP.

# 23.4 Distributed Array Processor (DAP)

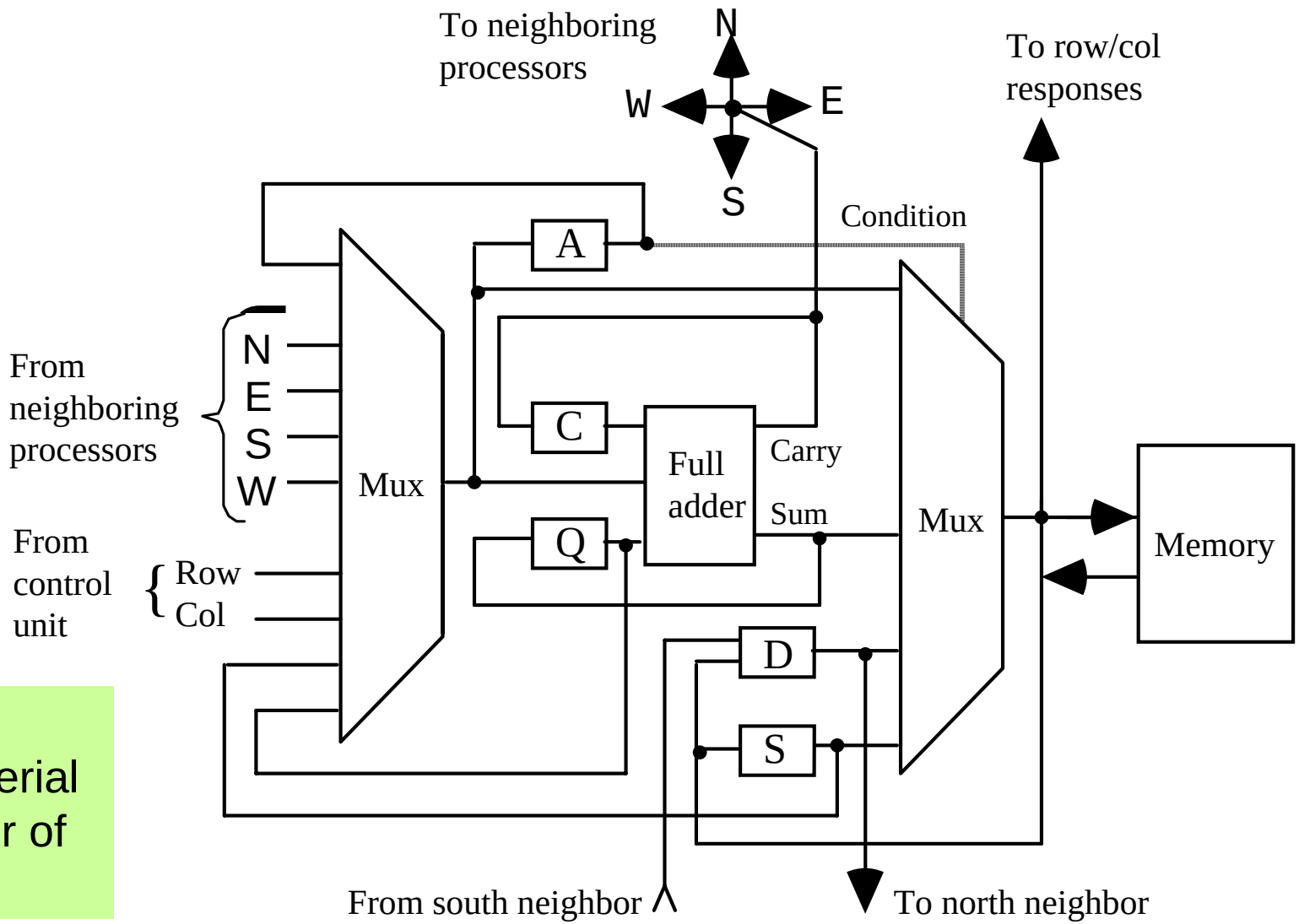



Fig. 23.6
The bit-serial processor of DAP.

# DAP's High-Level Structure

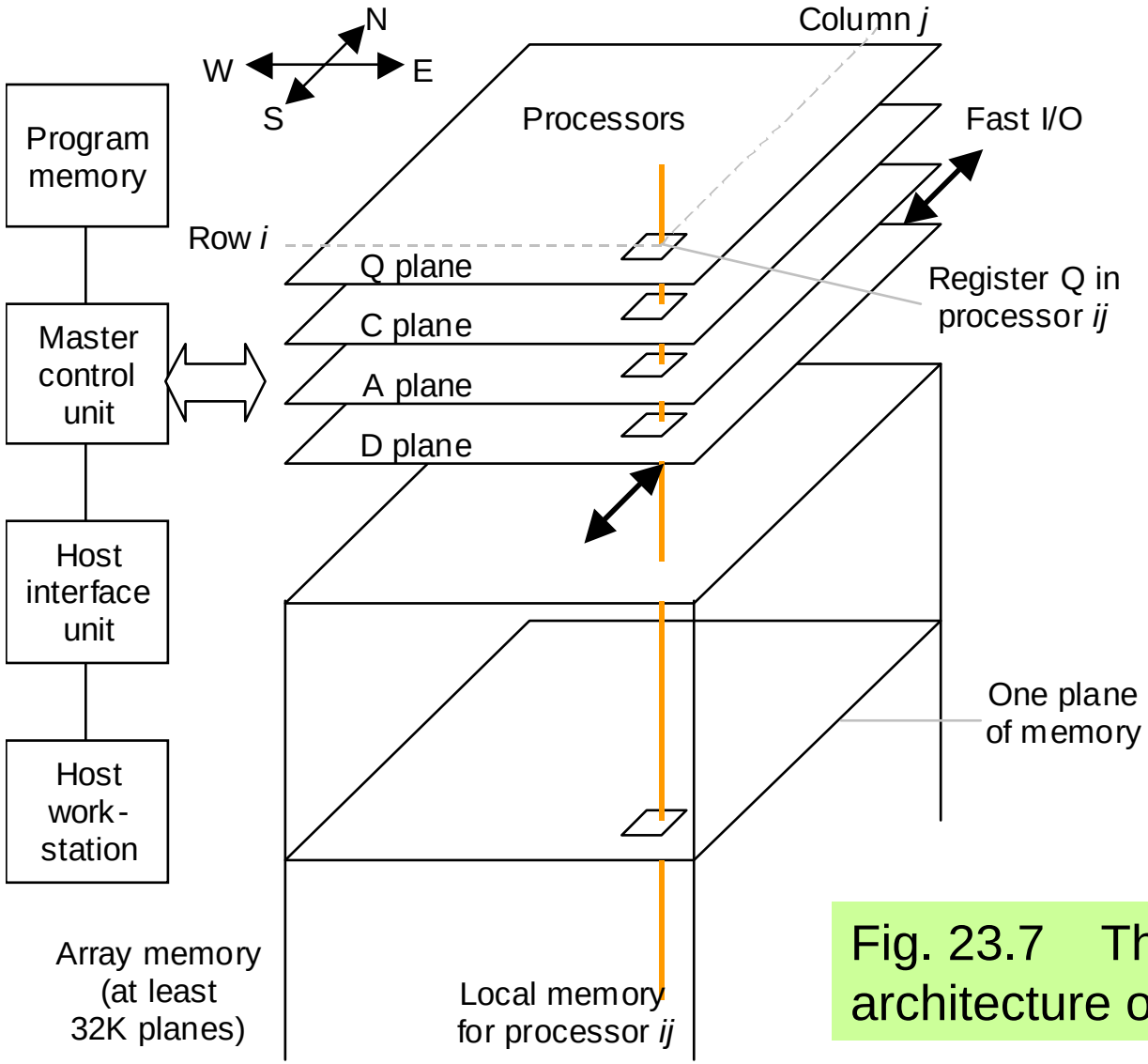



Fig. 23.7 The high-level architecture of DAP system.

# 23.5 Hypercubic Connection Machine 2

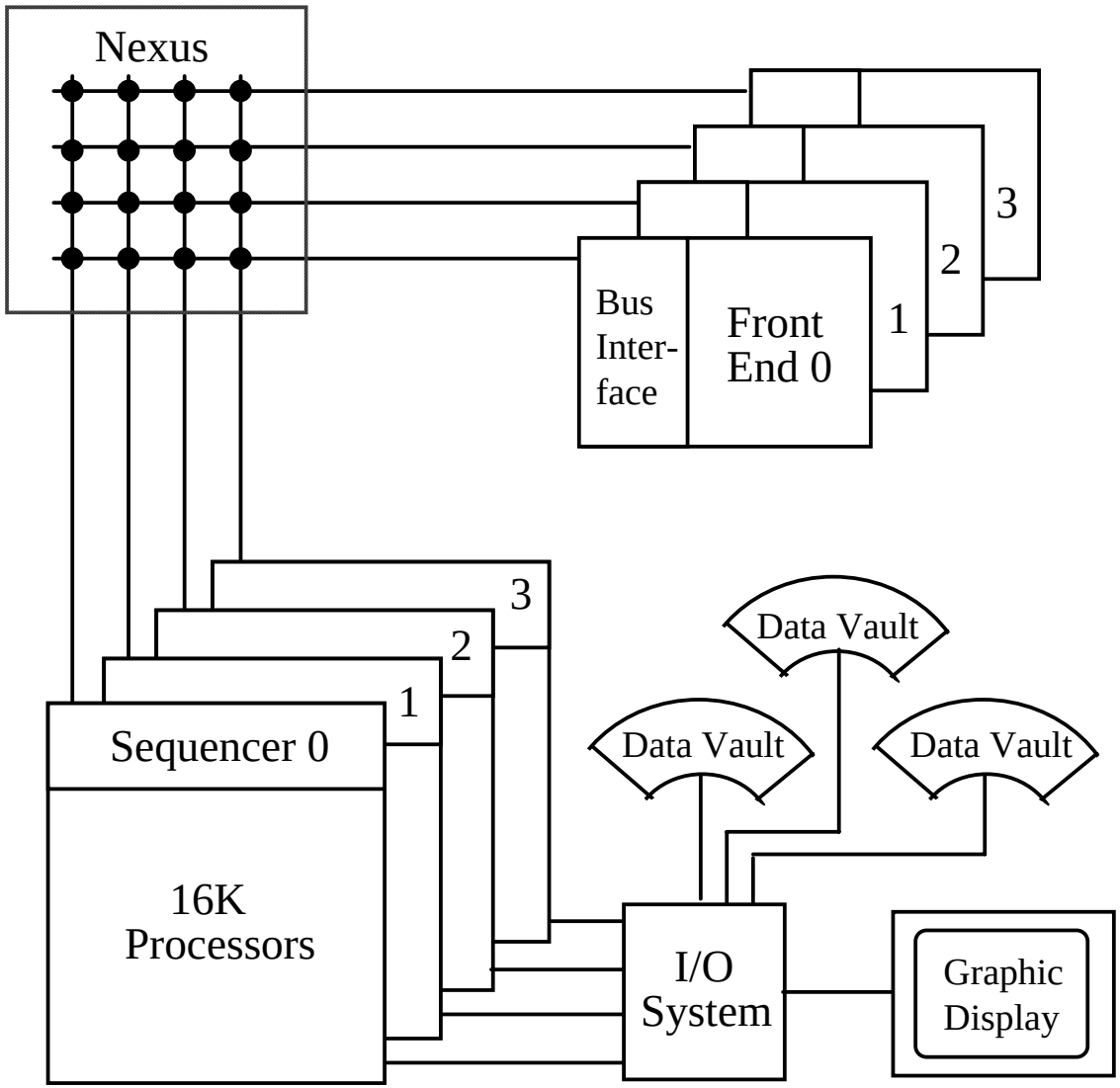



Fig. 23.8
The architecture of CM-2.

# The Simple Processing Elements of CM-2

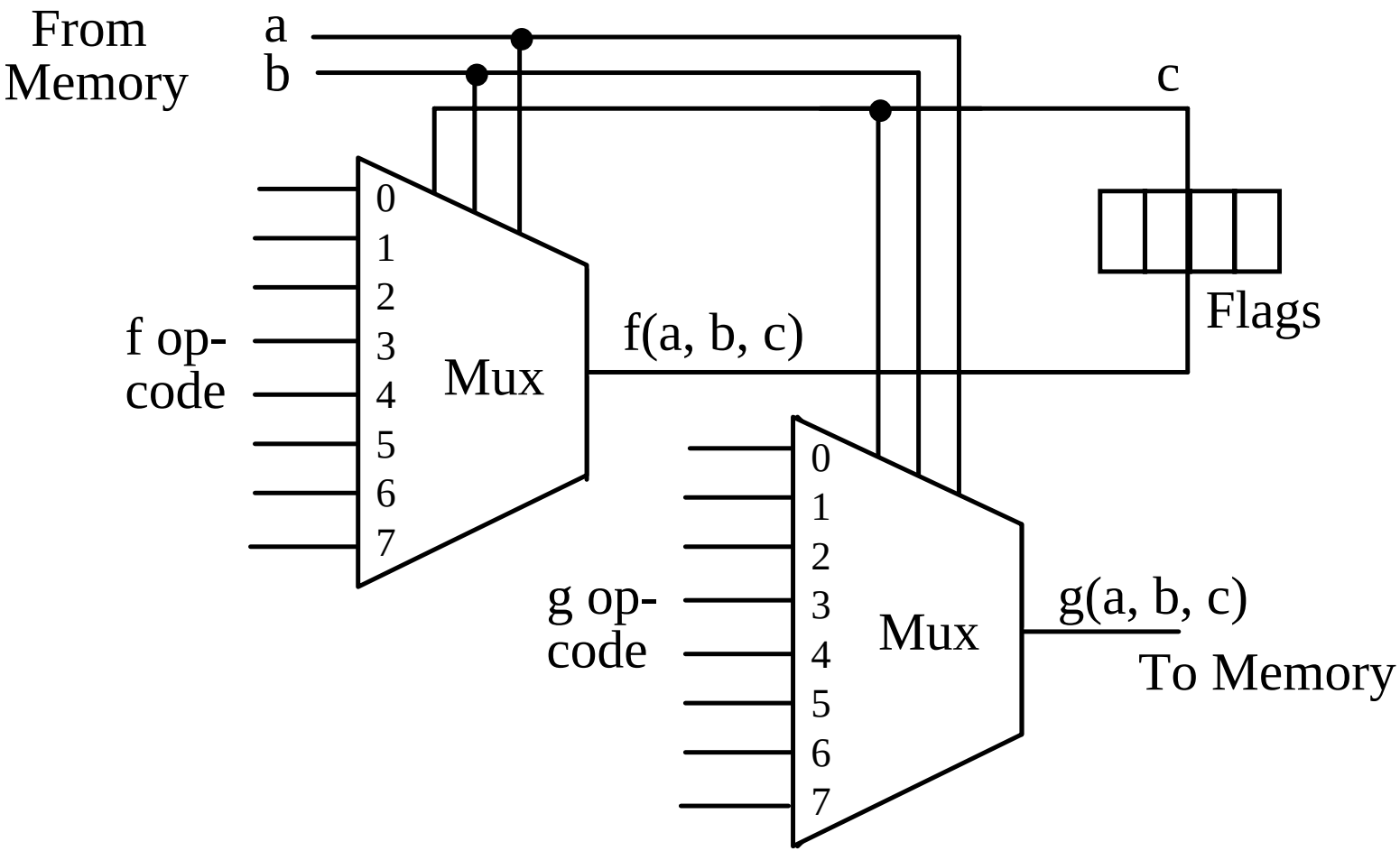



Fig. 23.9 The bit-serial ALU of CM-2.

# 23.6 Multiconnected MasPar MP-2

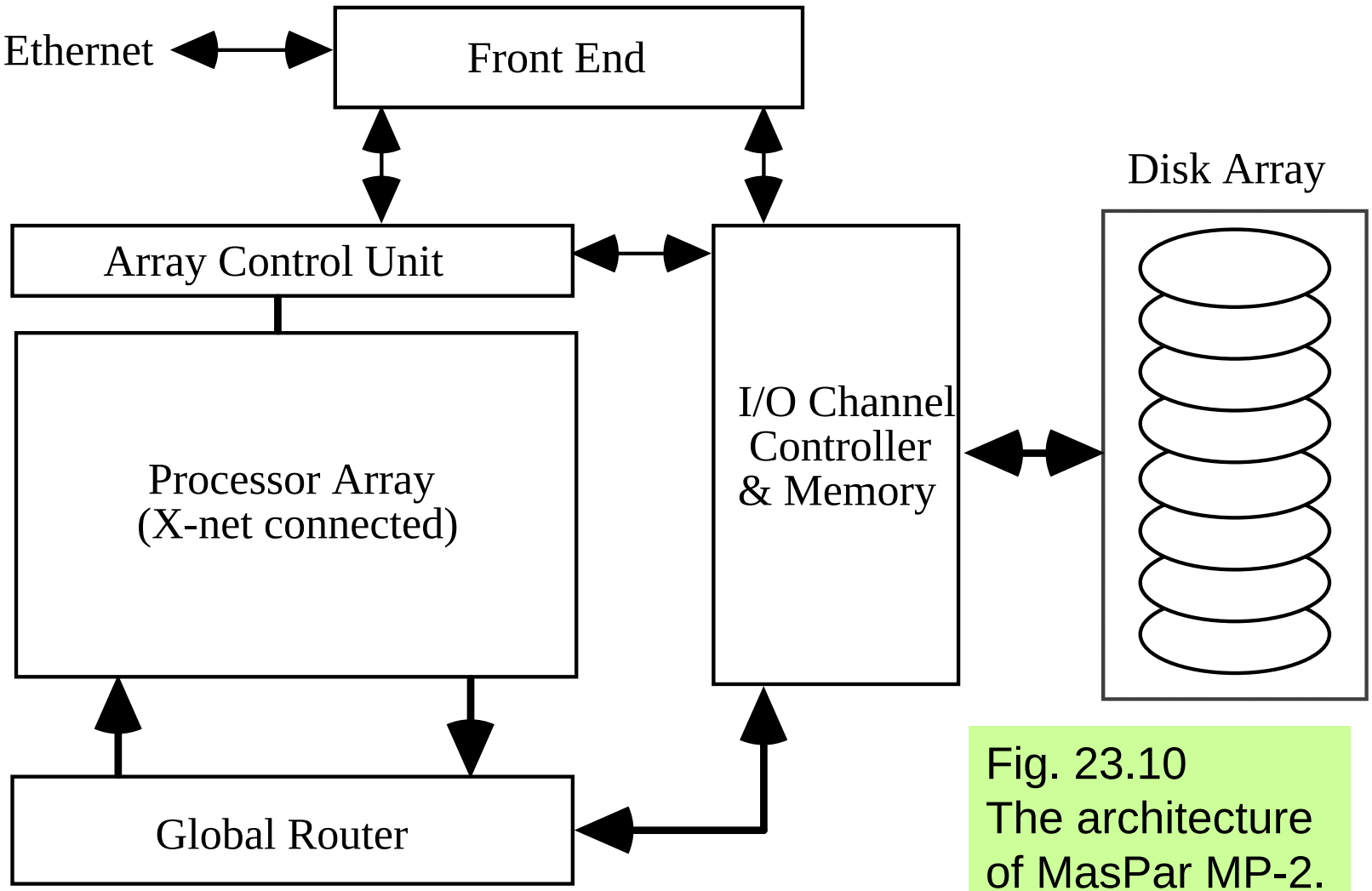



Fig. 23.10
The architecture
of MasPar MP-2.

# The Interconnection Network of MasPar MP-2

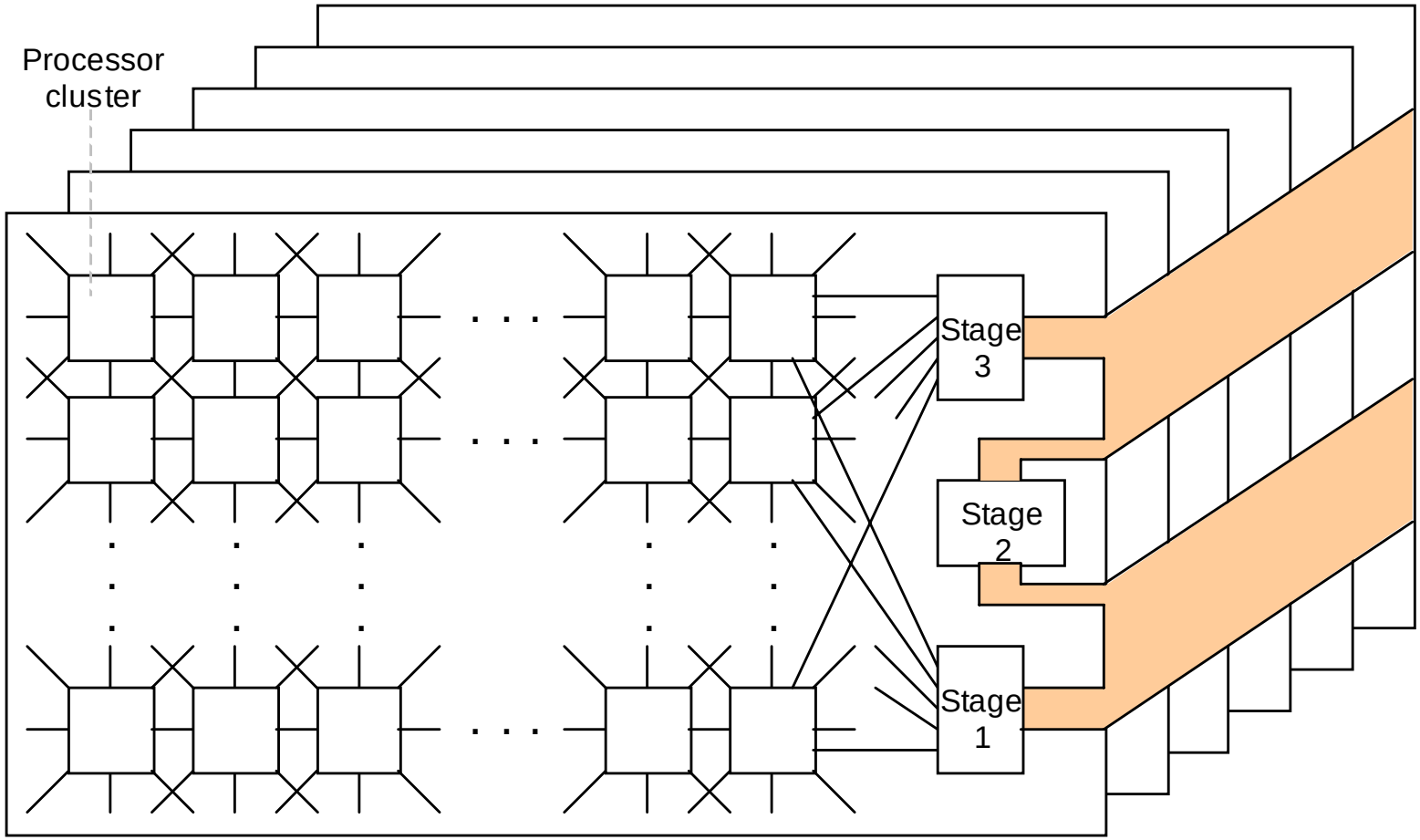



Fig. 23.11 The physical packaging of processor clusters and the 3-stage global router in MasPar MP-2 .

# The Processing Nodes of MasPar MP-2

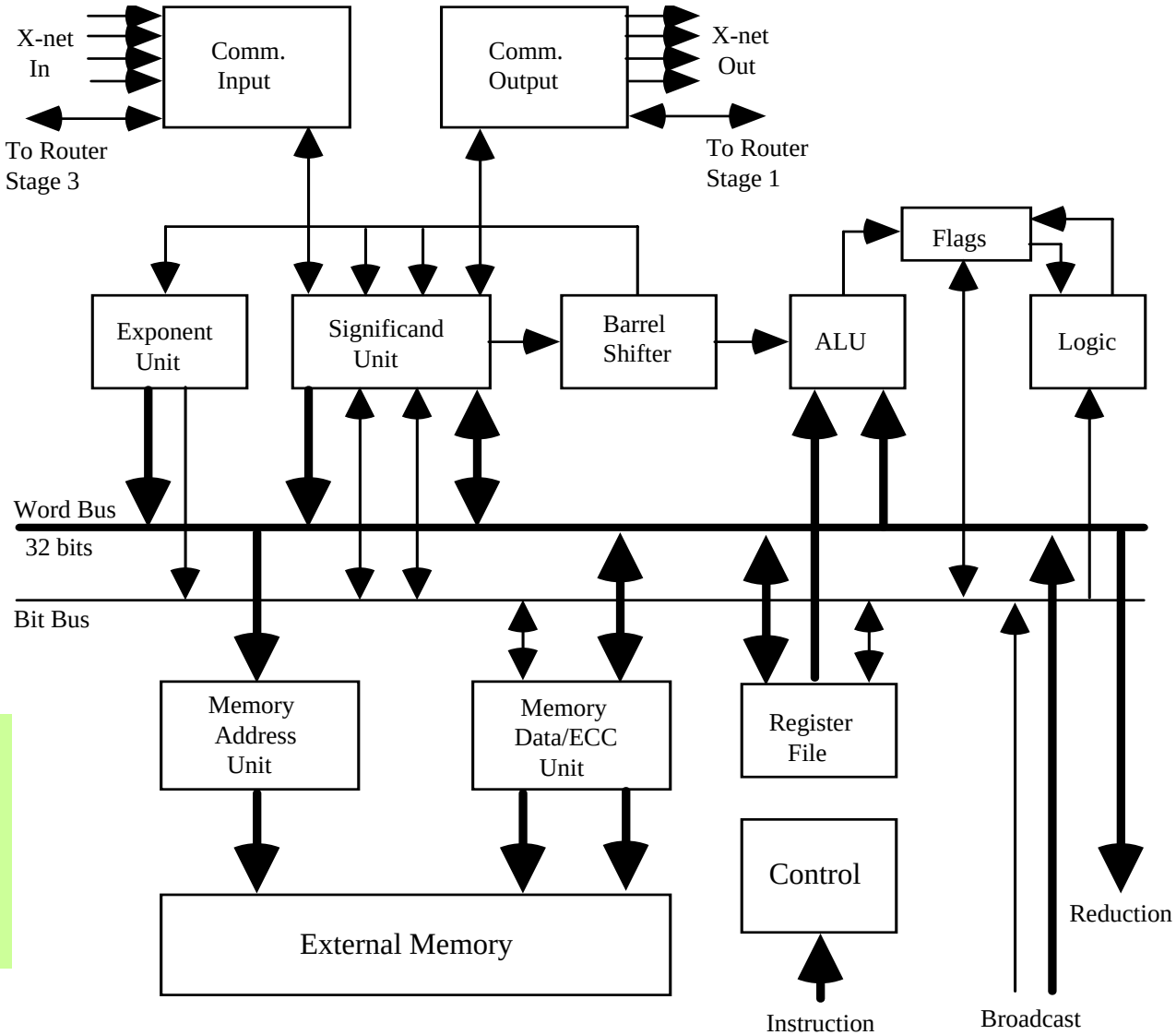



Fig. 23.12
Processor
architecture in
MasPar MP-2.

# 24 Past, Present, and Future

Put the state of the art in context of history and future trends:

- Quest for higher performance, and where it is leading us
- Interplay between technology progress and architecture

| Topics in This Chapter | |
|---|---|
| 24.1 | Milestones in Parallel Processing |
| 24.2 | Current Status, Issues, and Debates |
| 24.3 | TFLOPS, PFLOPS, and Beyond |
| 24.4 | Processor and Memory Technologies |
| 24.5 | Interconnection Technologies |
| 24.6 | The Future of Parallel Processing |

# 24.1 Milestones in Parallel Processing

**1840s** Desirability of computing multiple values at the same time was noted in connection with Babbage's Analytical Engine

**1950s** von Neumann's and Holland's cellular computers were proposed

**1960s** The ILLIAC IV SIMD machine was designed at U Illinois based on the earlier SOLOMON computer

**1970s** Vector machines, with deeply pipelined function units, emerged and quickly dominated the supercomputer market; early experiments with shared memory multiprocessors, message-passing multicomputers, and gracefully degrading systems paved the way for further progress

**1980s** Fairly low-cost and massively parallel supercomputers, using hypercube and related interconnection schemes, became available

**1990s** Massively parallel computers with mesh or torus interconnection, and using wormhole routing, became dominant

**2000s** Commodity and network-based parallel computing took hold, offering speed, flexibility, and reliability for server farms and other areas

# 24.2 Current Status, Issues, and Debates

| **Design choices and key debates:** | **Related user concerns:** |
|---|---|
| **Architecture** | |
| General- or special-purpose system? | Cost per MIPS |
| SIMD, MIMD, or hybrid? | Scalability, longevity |
| **Interconnection** | |
| Shared-medium, direct, or multistage? | Cost per MB/s |
| Custom or commodity? | Expandability, reliability |
| **Routing** | |
| Oblivious or adaptive? | Speed, efficiency |
| Wormhole, packet, or virtual cut-through? | Overhead, saturation limit |
| **Programming** | |
| Shared memory or message passing? | Ease of programming |
| New languages or libraries? | Software portability |

# 24.3 TFLOPS, PFLOPS, and Beyond

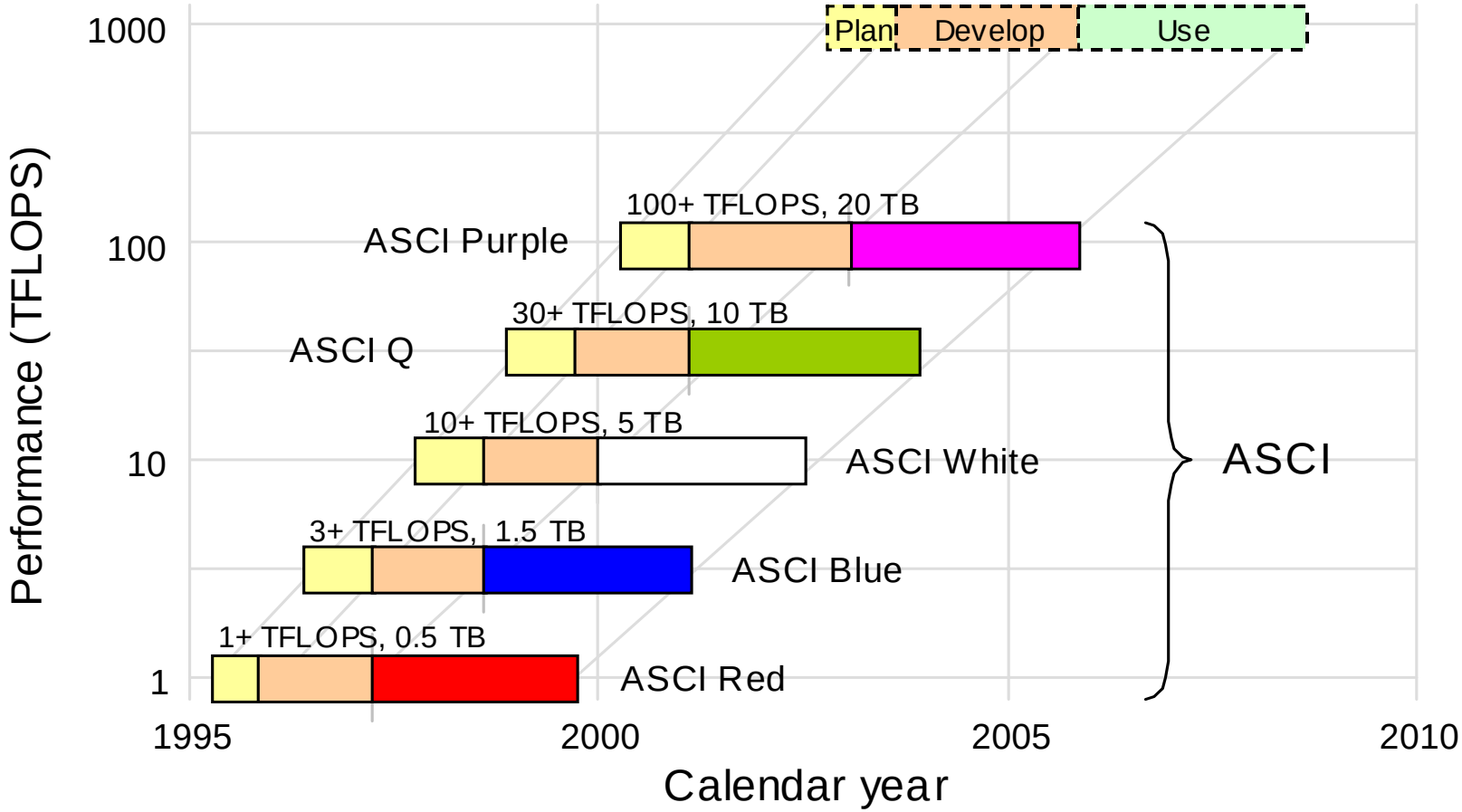
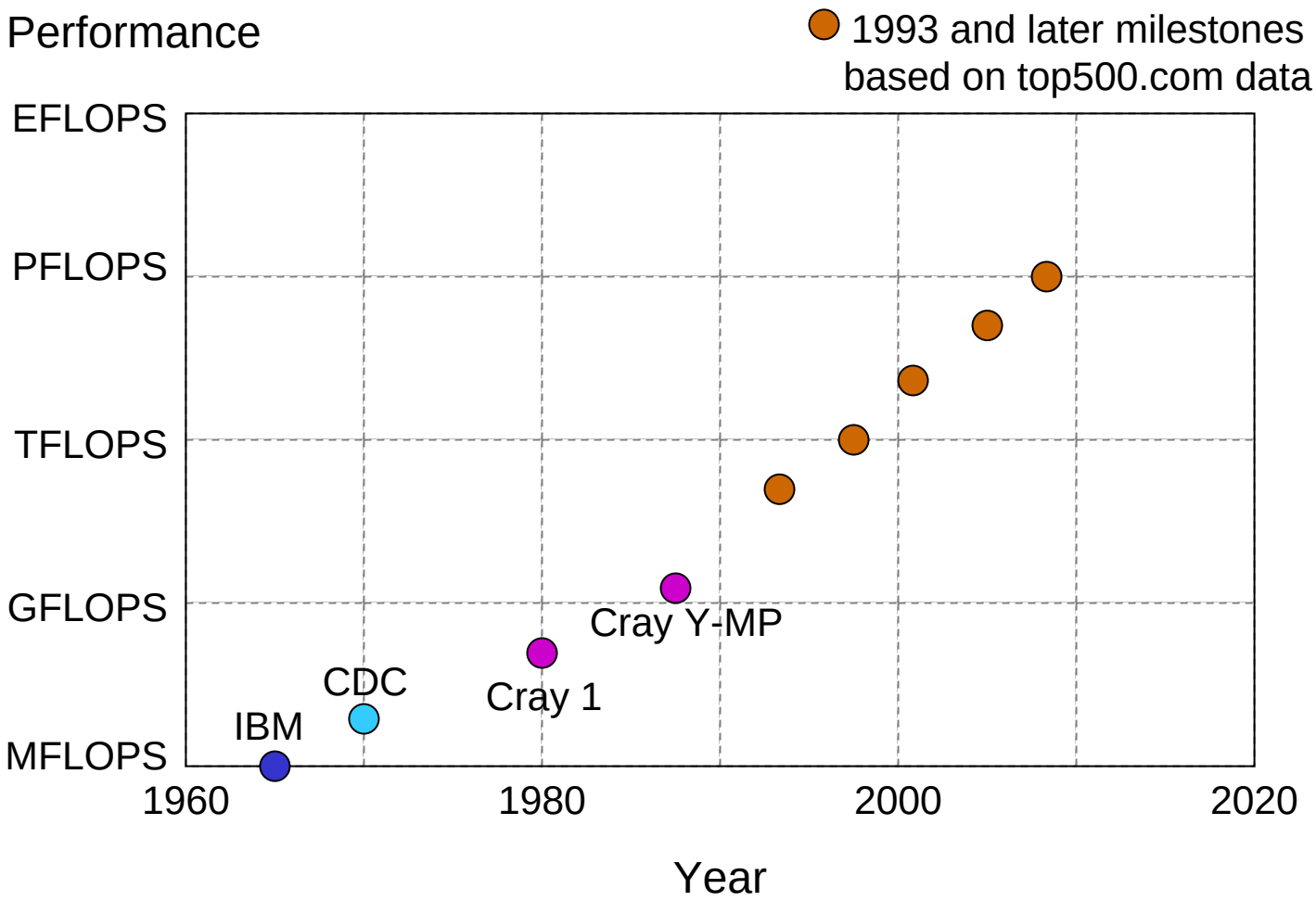



Fig. 24.1 Milestones in the Accelerated Strategic Computing Initiative (ASCI) program, sponsored by the US Department of Energy, with extrapolation up to the PFLOPS level.

# Performance Milestones in Supercomputers



# 24.4 Processor and Memory Technologies

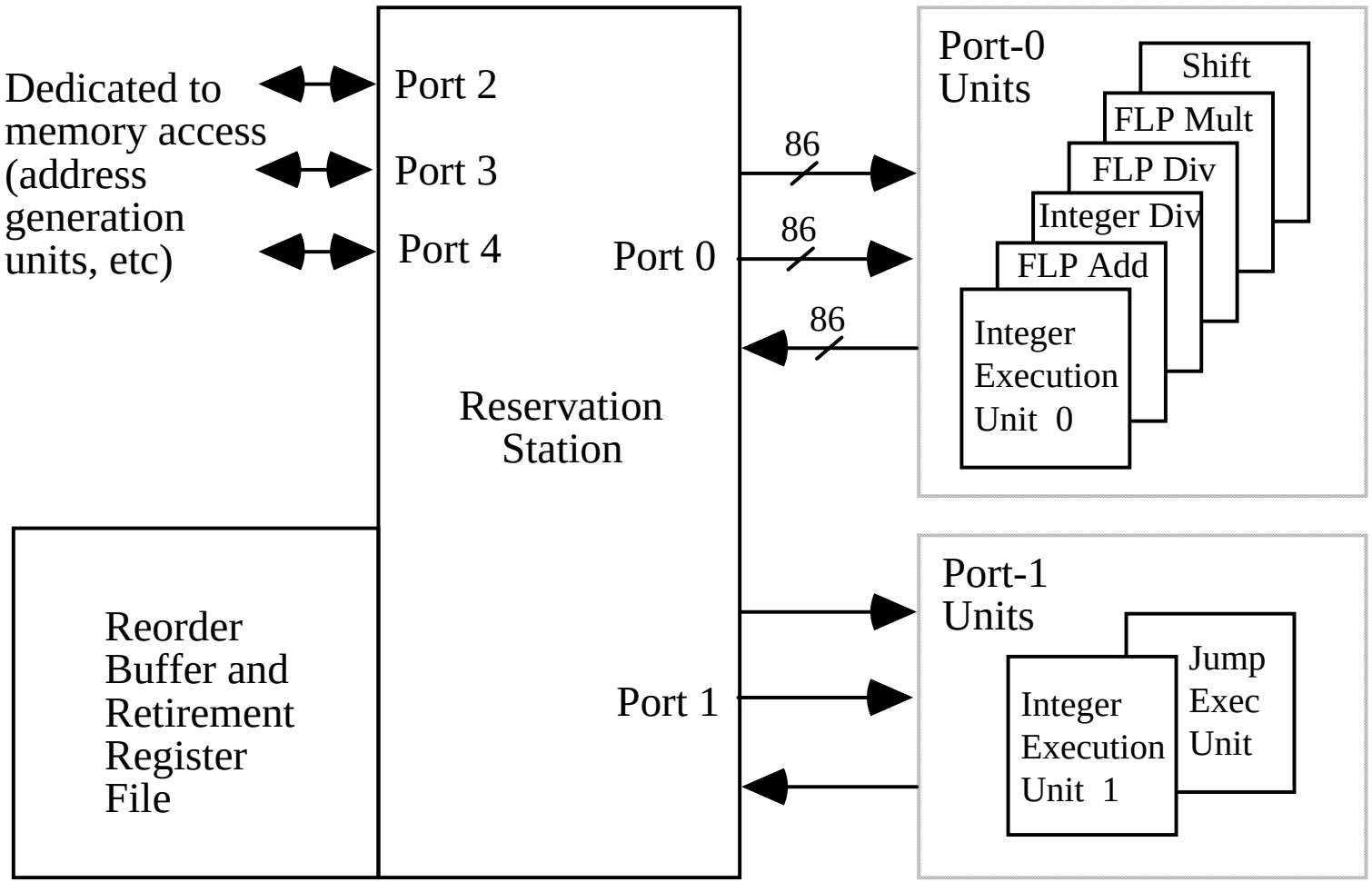



Fig. 24.2 Key parts of the CPU in the Intel Pentium Pro microprocessor.

# 24.5 Interconnection Technologies

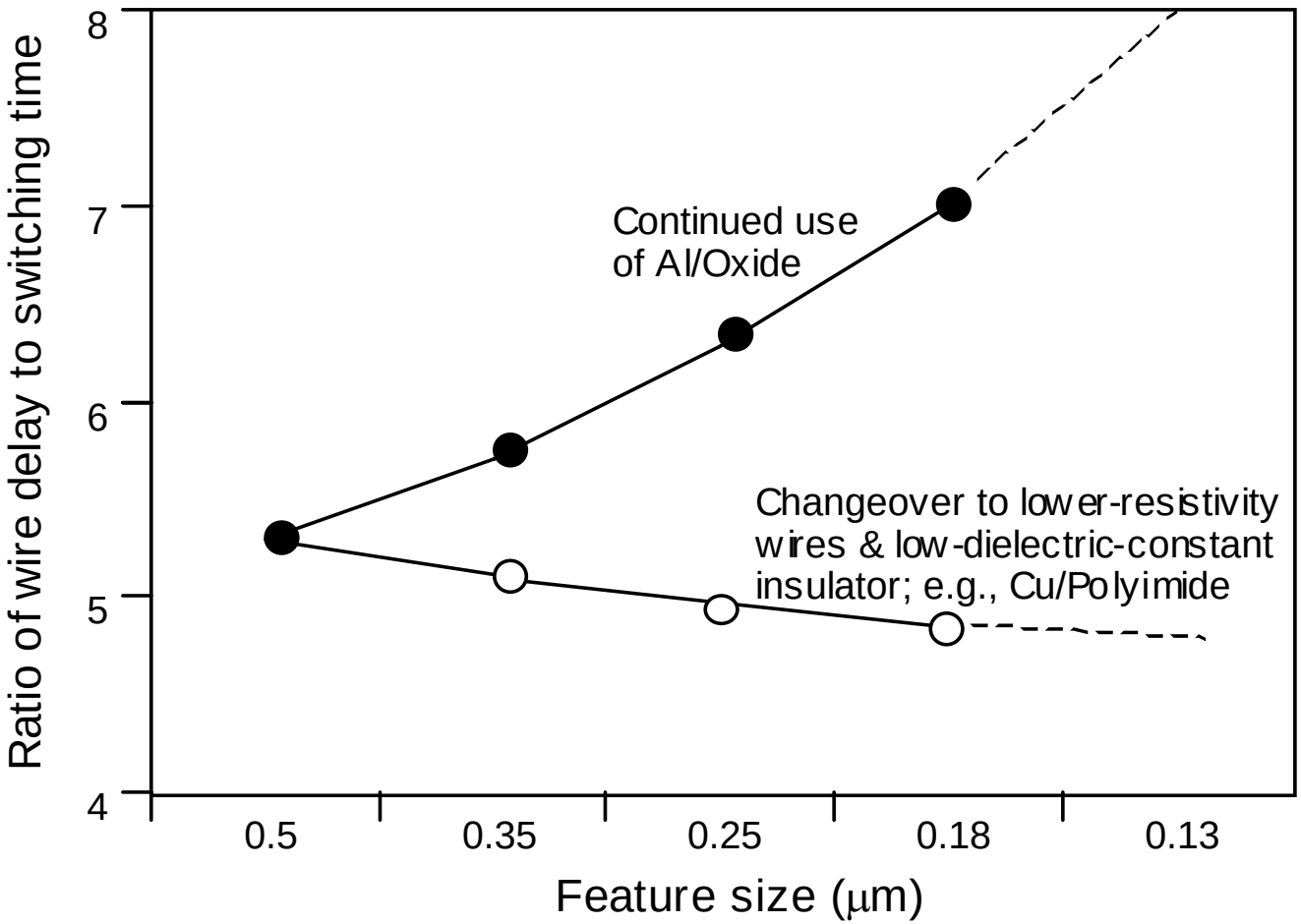



Fig. 24.3 Changes in the ratio of a 1-cm wire delay to device switching time as the feature size is reduced.

# Intermodule and Intersystem Connection Options

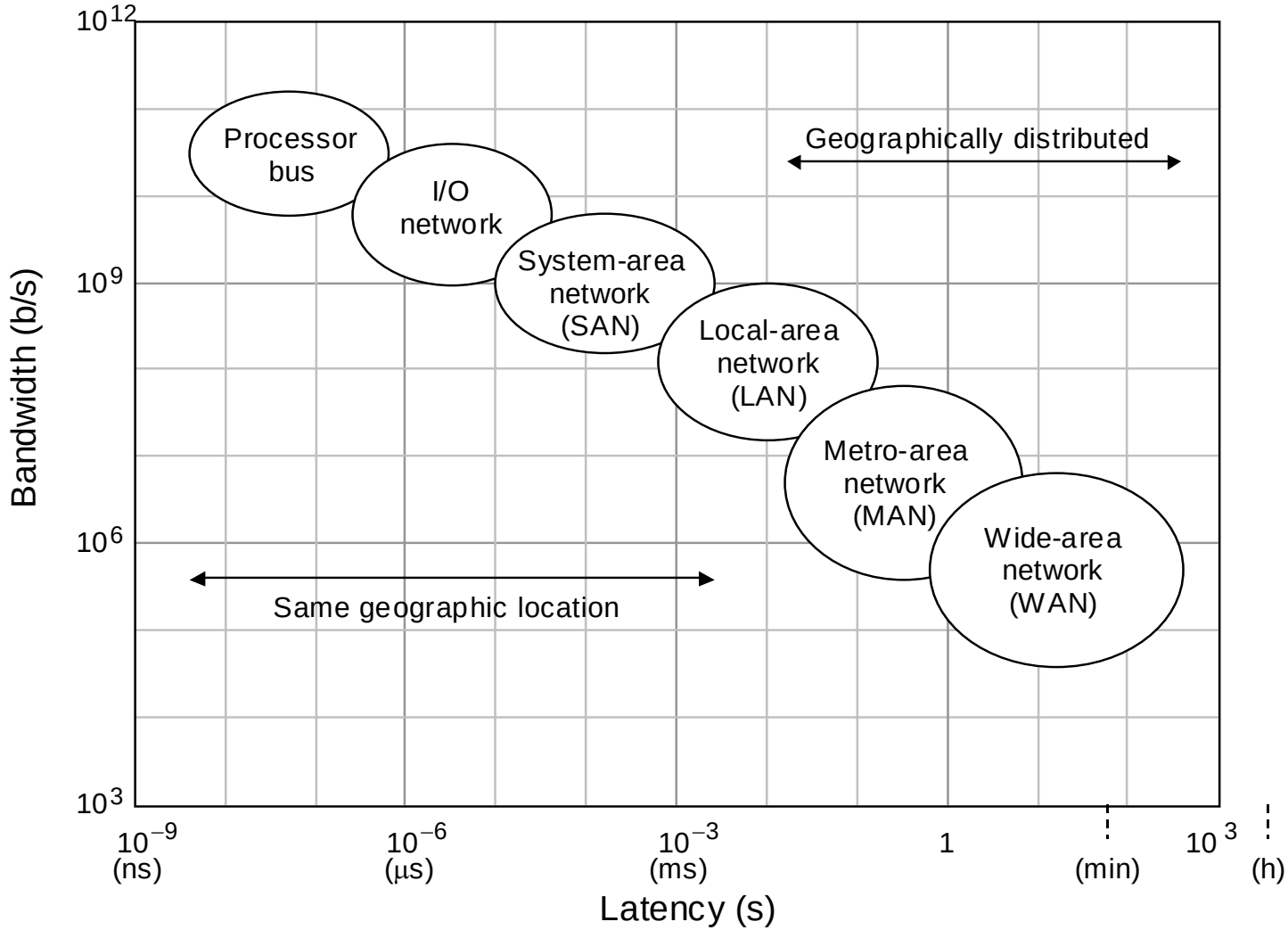
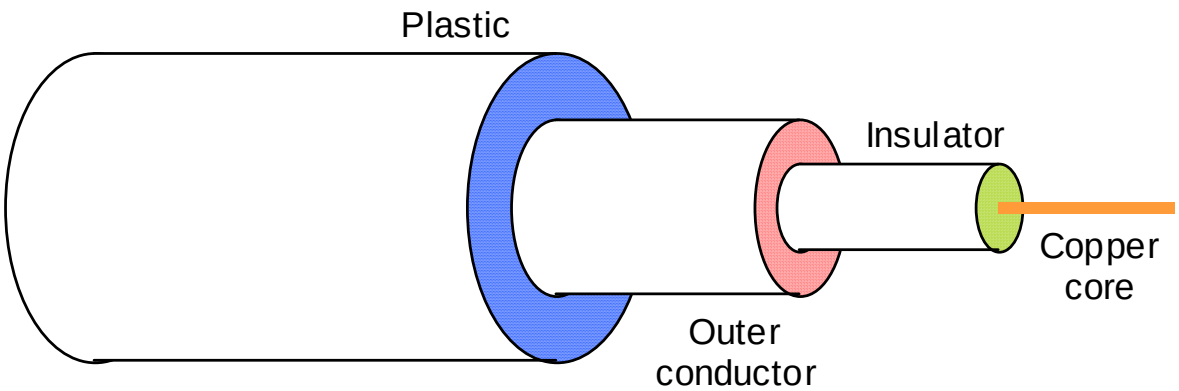


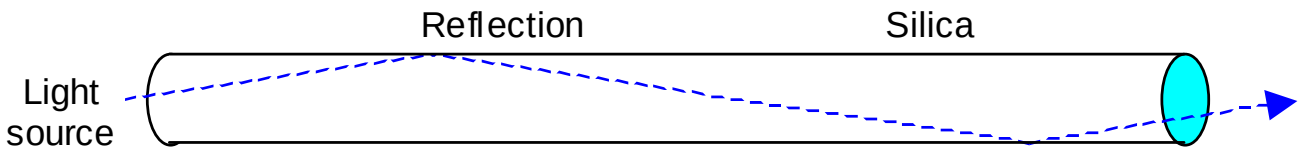

Fig. 24.4 Various types of intermodule and intersystem connections.

# System Interconnection Media

Twisted pair




Coaxial cable




Optical fiber

Fig. 24.5 The three commonly used media for computer and network connections.

# 24.6 The Future of Parallel Processing

In the early 1990s, it was unclear whether the TFLOPS performance milestone would be reached with a thousand or so GFLOPS nodes or with a million or so simpler MFLOPS processors

The eventual answer was closer to 1K × GFLOPS than 1M × MFLOPS PFLOPS performance was achieved in more or less the same manner

**Asynchronous design** for its speed and greater energy economy: Because pure asynchronous circuits create some difficulties, designs that are locally synchronous and globally asynchronous are flourishing

**Intelligent memories** to help remove the memory wall: Memory chips have high internal bandwidths, but are limited by pins in how much data they can handle per clock cycle

**Reconfigurable systems** for adapting hardware to application needs: Raw hardware that can be configured via "programming" to adapt to application needs may offer the benefits of special-purpose processing

**Network and grid computing** to allow flexible, ubiquitous access: Resource of many small and large computers can be pooled together via a network, offering supercomputing capability to anyone

# Parallel Processing for Low-Power Systems

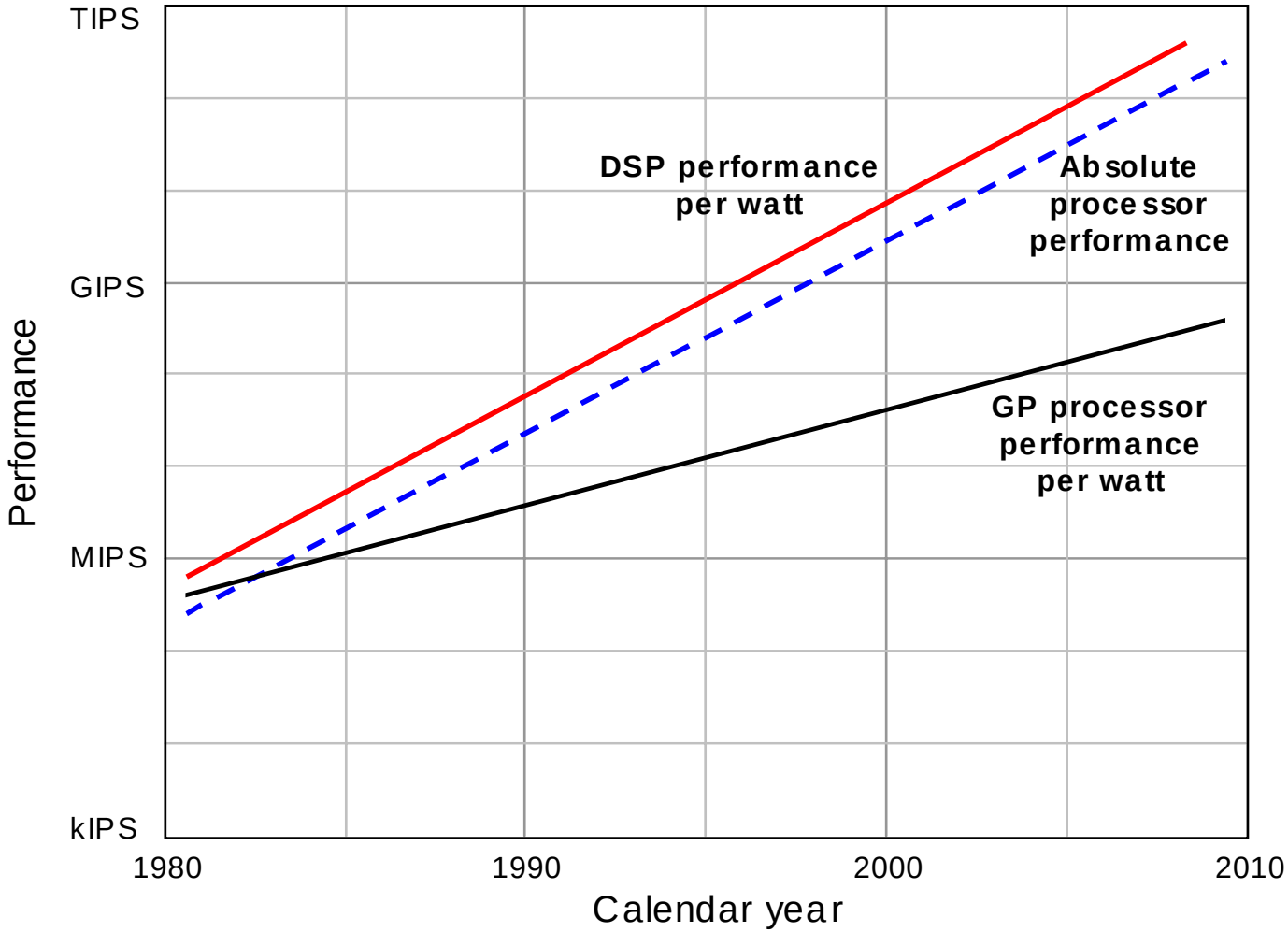
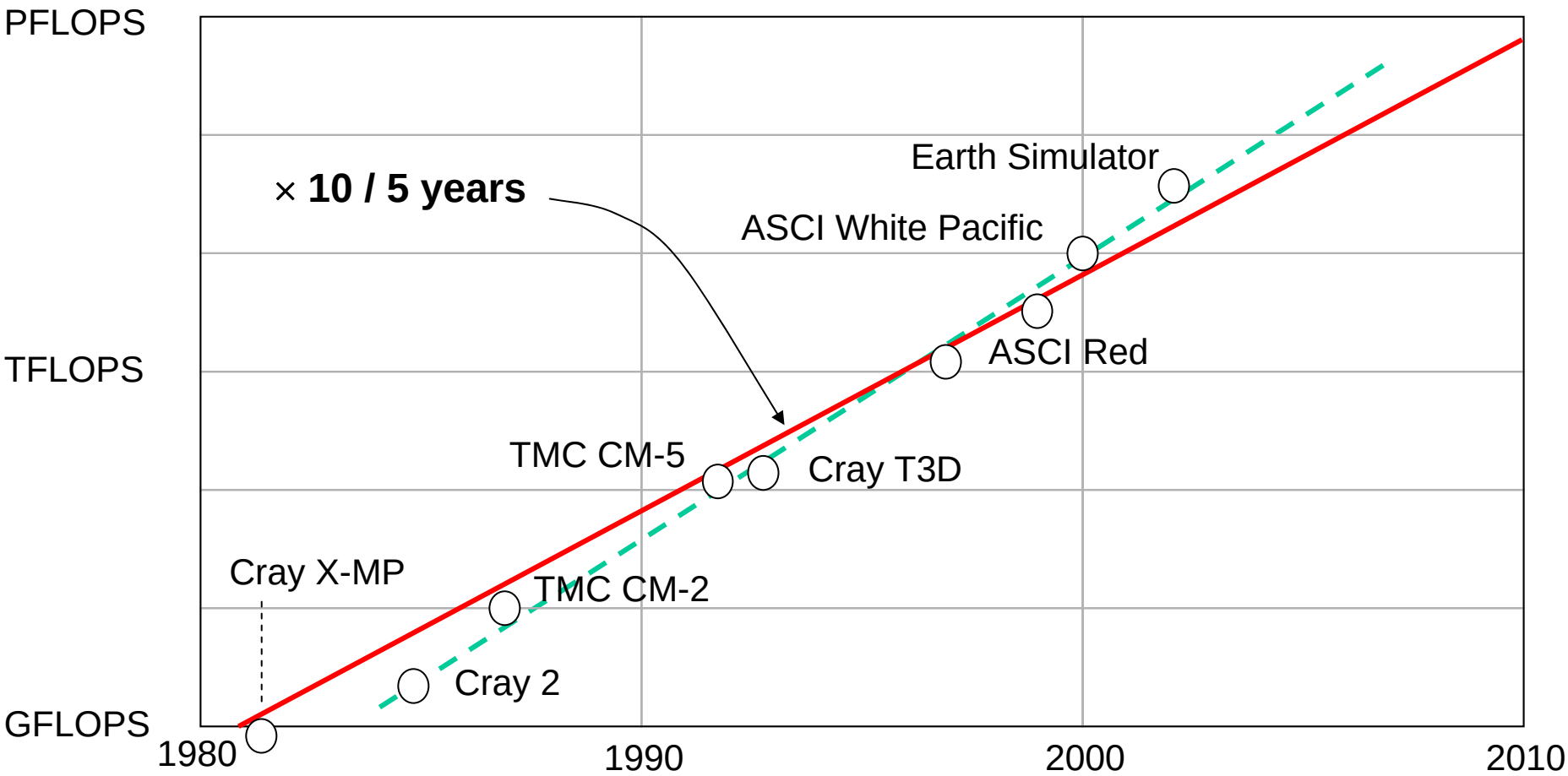


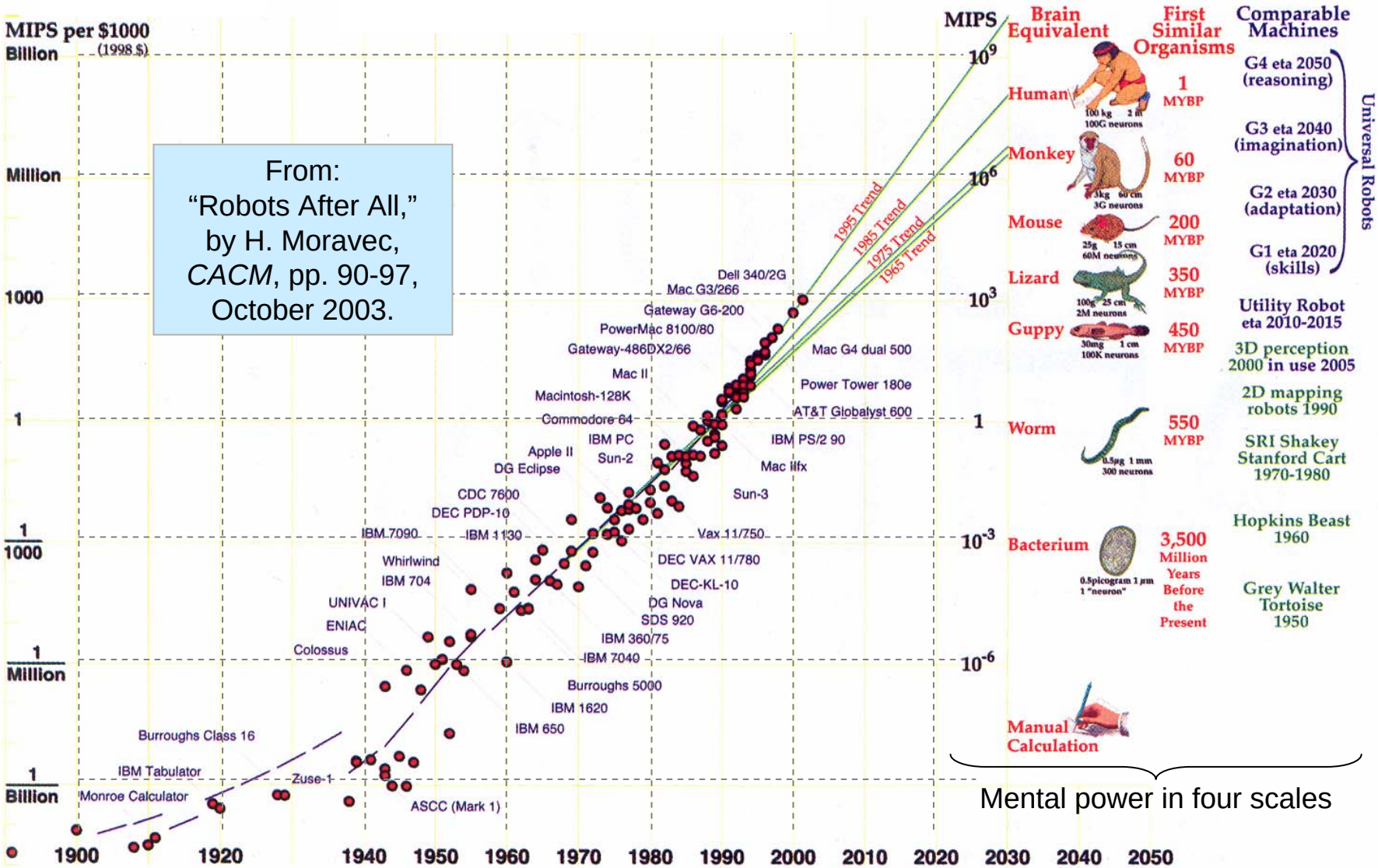

Figure 25.1 of Parhami's computer architecture textbook.

# Peak Performance of Supercomputers




Dongarra, J., "Trends in High Performance Computing," *Computer J.*, Vol. 47, No. 4, pp. 399-403, 2004. [Dong04]

# Evolution of Computer Performance/Cost



# Two Approaches to Parallel Processing

**Architecture (topology) and algorithm considered in tandem**

+ Potentially very high performance

– Large effort to develop efficient algorithms

**Users interact with topology-independent platforms**

+ Less steep learning curve

+ Program portability

– Sacrifice some performance for ease of use

Main examples:

Virtual shared memory

Message-passing interface

The topology-independent approach is currently dominant

# ABCs of Parallel Processing in One Slide

## A Amdahl's Law (Speedup Formula)

**Bad news –** Sequential overhead will kill you, because:

Speedup = $T_1/T_p \leq 1/[f + (1 - f)/p] \leq min(1/f, p)$

**Morale:** For $f = 0.1$, speedup is at best 10, regardless of peak OPS.

## B Brent's Scheduling Theorem

**Good news –** Optimal scheduling is very difficult, but even a naive scheduling algorithm can ensure:

$T_1/p \leq T_p < T_1/p + T_\infty = (T_1/p)[1 + p/(T_1/T_\infty)]$

**Result:** For a reasonably parallel task (large $T_1/T_\infty$), or for a suitably small $p$ (say, $p < T_1/T_\infty$), good speedup and efficiency are possible.

## C Cost-Effectiveness Adage

**Real news –** The most cost-effective parallel solution may not be the one with highest peak OPS (communication?), greatest speed-up (at what cost?), or best utilization (hardware busy doing what?).

**Analogy:** Mass transit might be more cost-effective than private cars even if it is slower and leads to many empty seats.

# Gordon Bell Prize for High-Performance Computing

**Established in 1988 by Gordon Bell, one of the pioneers in the field**

Prizes are given annually in several categories (there are small cash awards, but the prestige of winning is the main motivator)

The two key categories are as follows; others have varied over time

**Peak performance:** The entrant must convince the judges that the submitted program runs faster than any other comparable application

**Price/Performance:** The entrant must show that performance of the application divided by the list price of the smallest system needed to achieve the reported performance is better than that of any other entry

Different hardware/software combinations have won the competition

Supercomputer performance improvement has outpaced Moore's law; however, progress shows signs of slowing down due to lack of funding

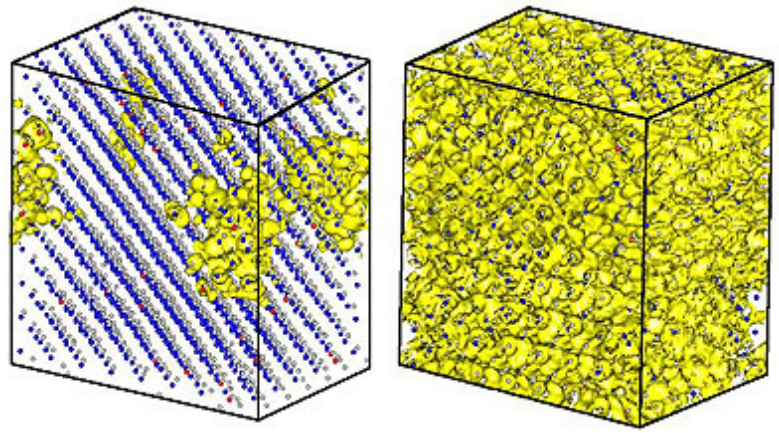
In past few years, improvements have been due to faster uniprocessors (2004 NRC Report *Getting up to Speed: The Future of Supercompuing*)

# Recent Gordon Bell Prizes, November 2008

**Performance in a Scientific Application: superconductor simulation**

Oak Ridge National Lab: Cray XT Jaguar, 150 000 cores, 1.35 PFLOPS


http://www.supercomputingonline.com/article.php?sid=16648



**Special Prize for Algorithm Innovation**

Lawrence Berkeley National Lab: harnessing potential of nanostructures

http://newscenter.lbl.gov/press-releases/2008/11/24/berkeley-lab-team-wins-special-acm-gordon-bell-prize-for-algorithm-innovation/