IO System

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

CP-226: Computer Architecture

Lecture 25 (24 April 2013)

CADSL

Typical Collection of I/O Devices

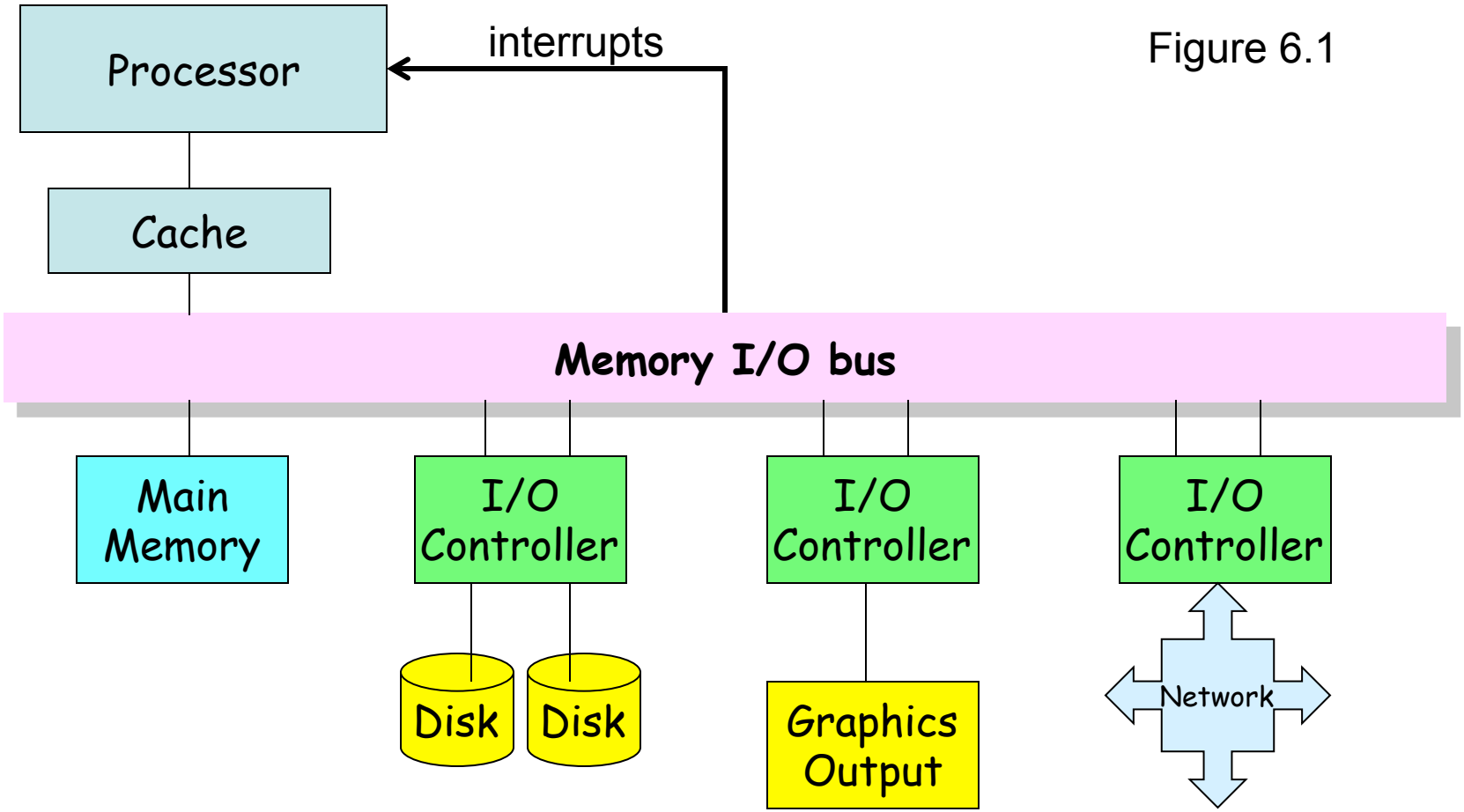


Figure 6.1

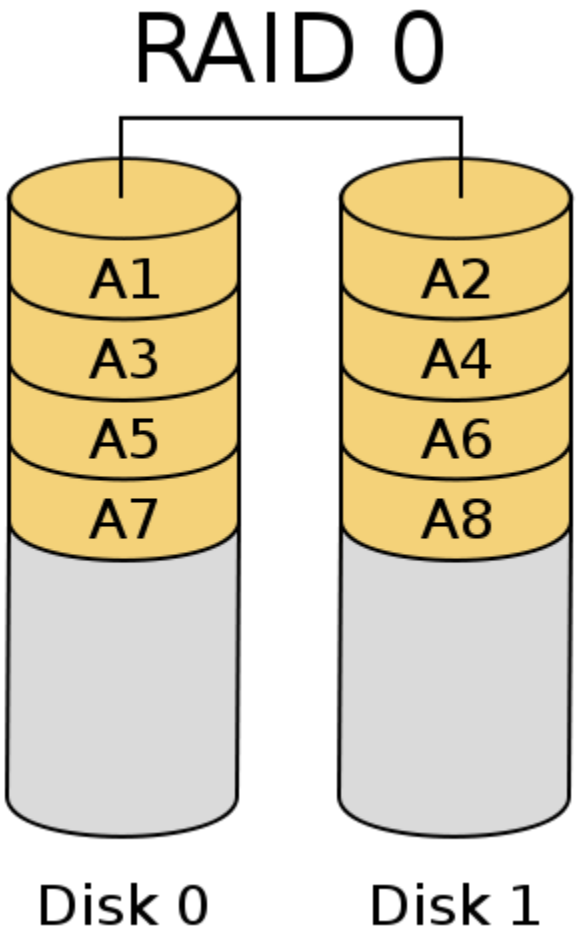
Types of Storage

Type	Size	Speed	Cost/bit
Register	< 1KB	< 1ns	$$$$
On-chip SRAM	8KB-6MB	< 10ns	$$$
Off-chip SRAM	1Mb – 16Mb	< 20ns	$$
DRAM	64MB – 1TB	< 100ns	$
Flash	64MB – 32GB	< 100us	~c
Disk	40GB – 1PB	< 20ms	~0

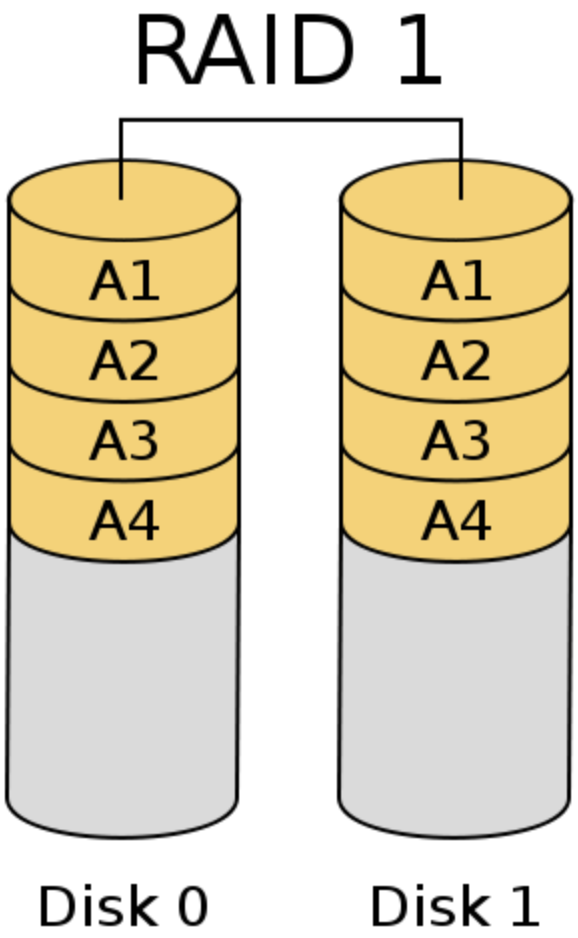
RAID: Redundant Array of Inexpensive Disks

- Dependability
  - Reliability: Measured by MTTF (mean time to failure)
  - Service interruption: measured by MTTR (mean time to repair)
  - Availability = MTTF/(MTTF + MTTR)
- What if we need 100 disks for storage?
- MTTF = 5 years / 100 = 18 days!
- RAID 0
  - Data striped (spread over multiple disks), but no error protection
- RAID 1
  - Mirror = stored twice = 100% overhead (most expensive)
- RAID 5
  - Block-wise parity = small overhead and small writes

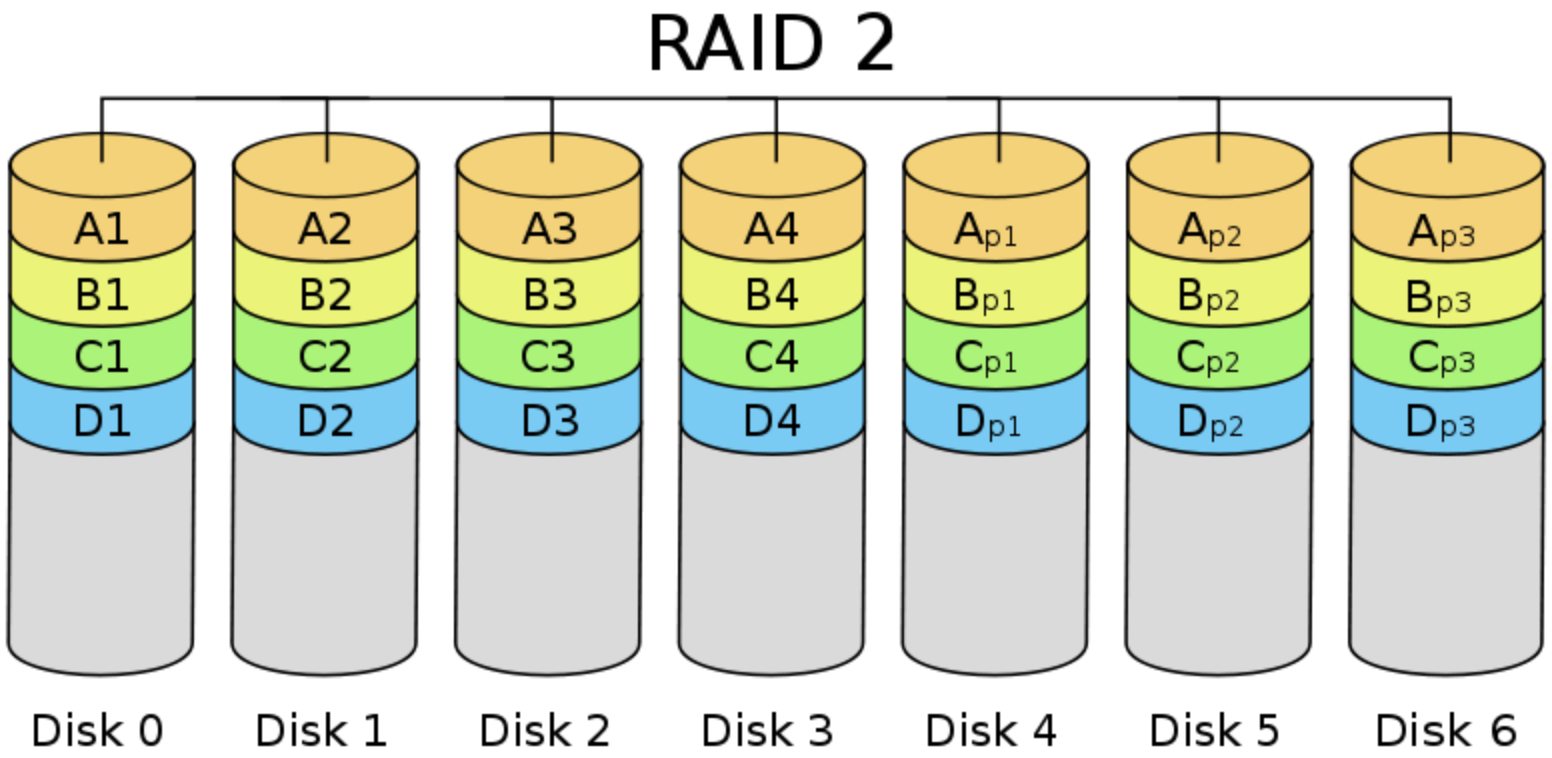
RAID 0: Block-level stripping



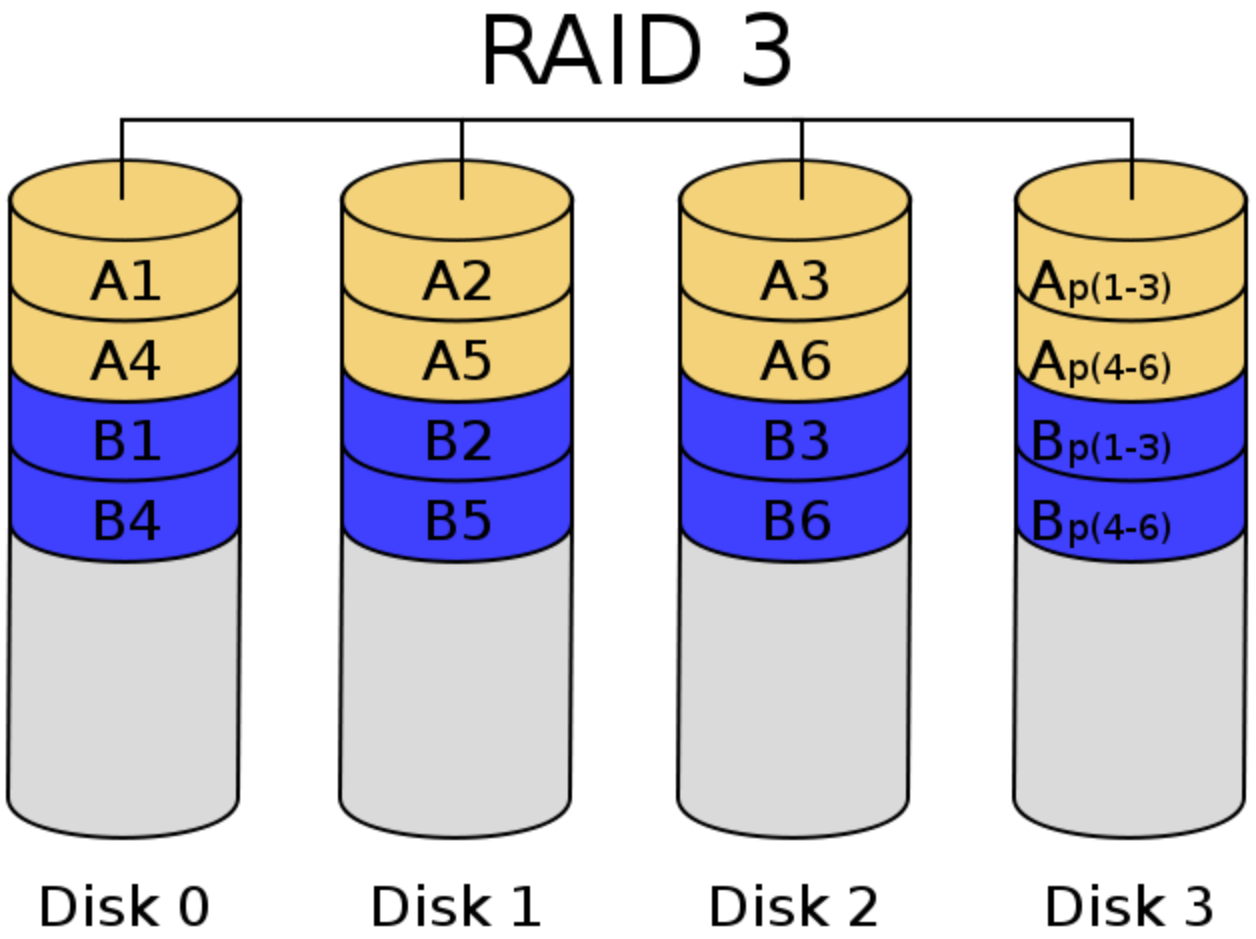
RAID 1: Mirroring



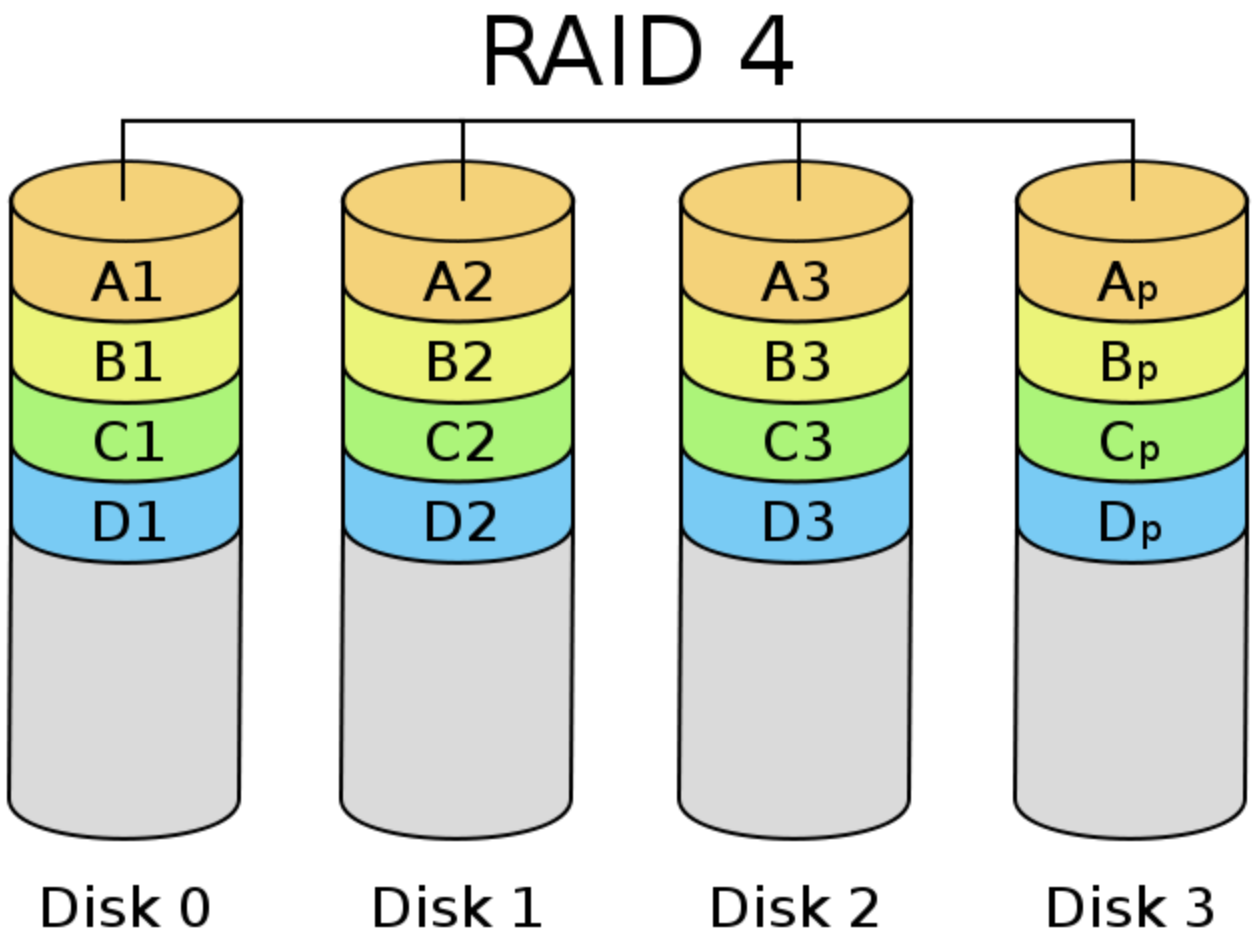
RAID 2: Bit-level Interleave with ECC



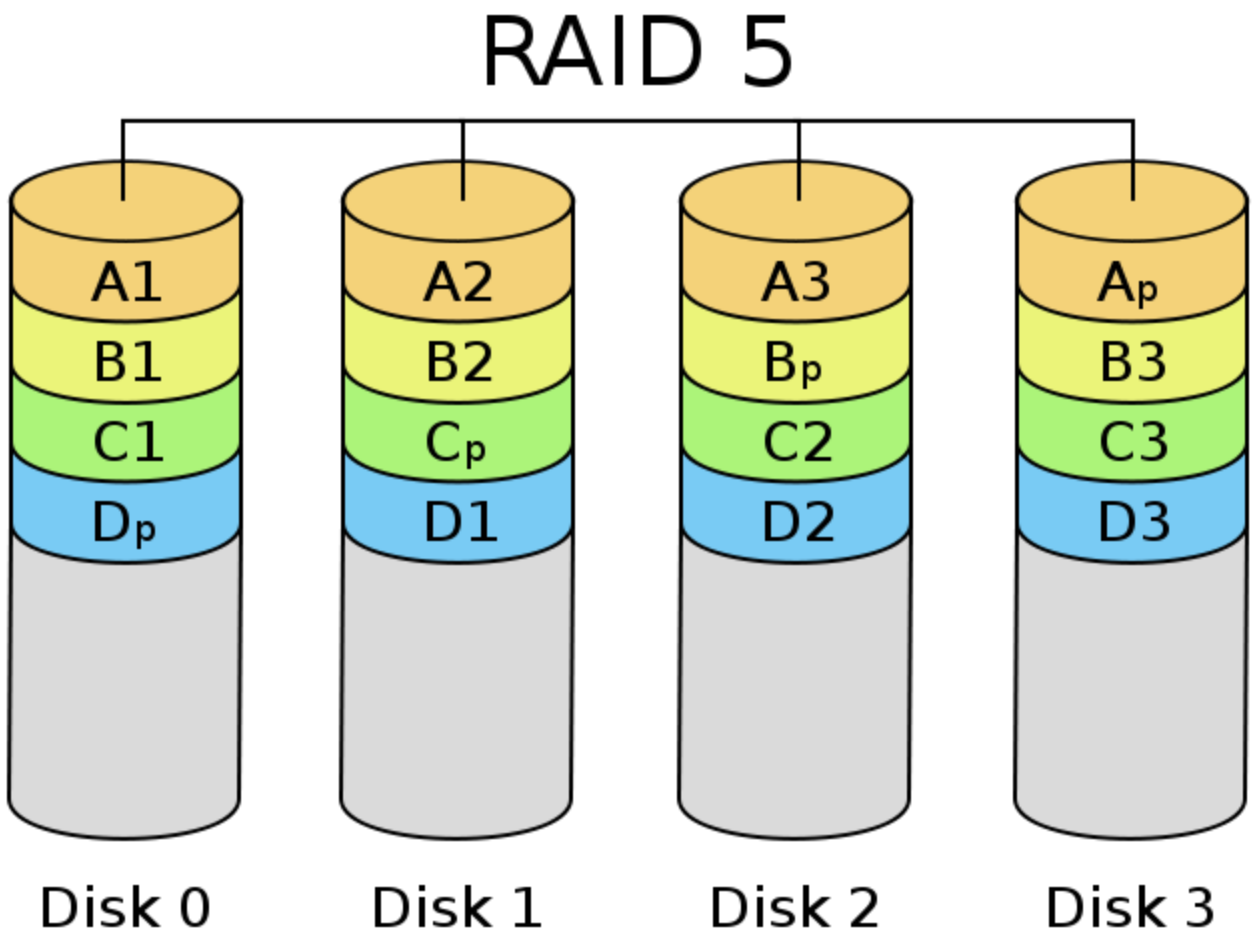
RAID 3: Byte-level Interleave



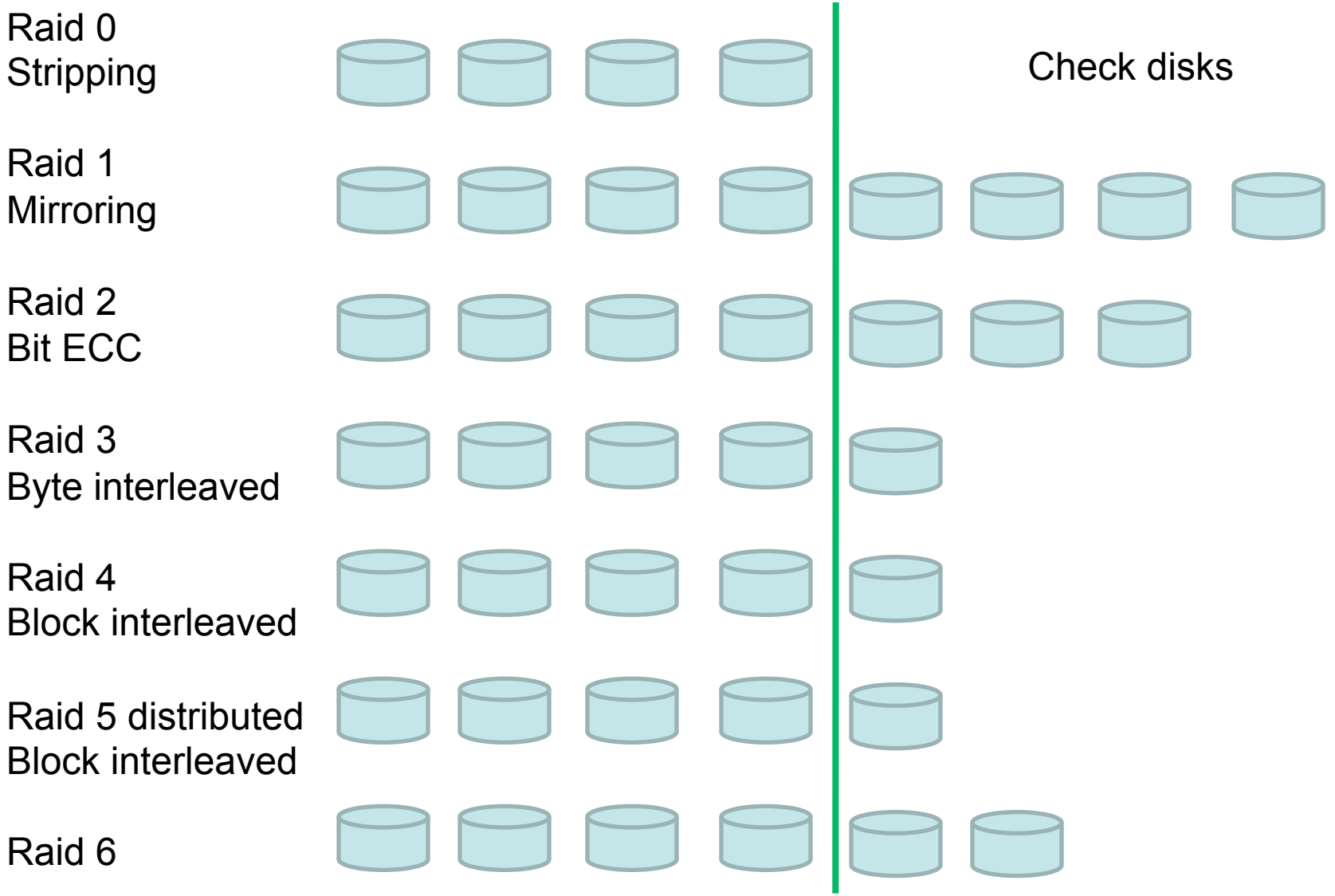
RAID 4: Block-level Interleave



RAID 5: Block-level interleaved distributed parity



RAID Illustrations



Tape

- Helical scan
  - 8mm video tape + ECC
  - 7GB/tape at $6/tape = <$1/GB
    - Note similar to cheap IDE hard drives!
  - Tape robots
- E.g. Library of Congress is 10TB text
  - 1500 tapes x $6 = $9000
  - Of course, not that simple

LAN = Ethernet

- Original Ethernet
  - One-write bus with collisions and exponential back-off
  - Within building
  - 10Mb/s (~= 1MB/s)
- Now Ethernet is
  - Point to point clients (switched network)
  - Client s/w, protocol unchanged
  - 100Mb/s => 1Gb/s

LAN

- Ethernet not technically optimal
  - 80x86 is not technically optimal either!
- Nevertheless, many efforts to displace it have failed (token ring, ATM)
- Emerging: System Area Network (SAN)
  - Reduce SW stack (TCP/IP processing)
  - Reduce HW stack (interface on memory bus)
  - Standard: Infiniband (http://www.infinibandta.org)

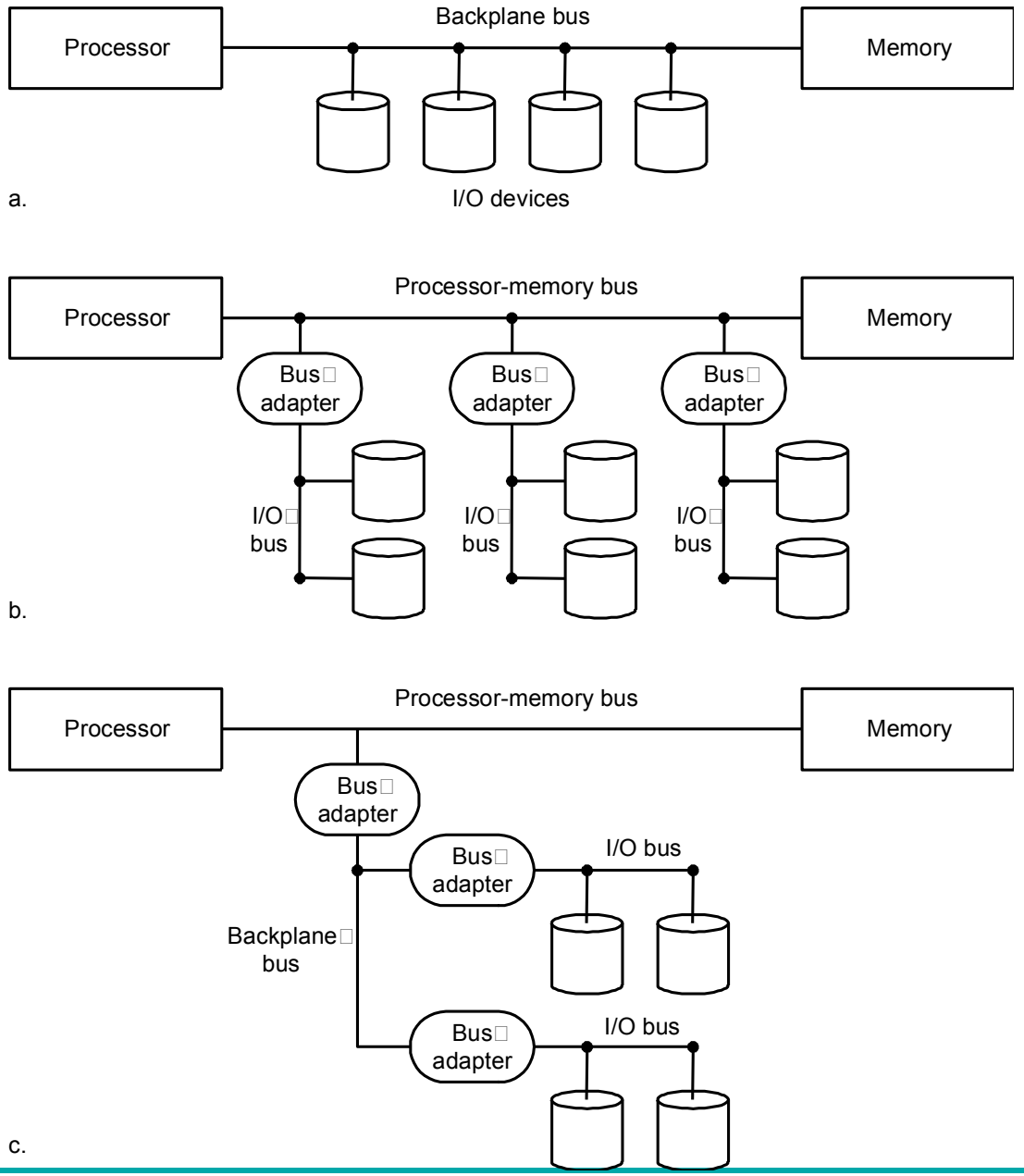
Bus

- A shared communication link between processor and memory, and I/O devices
- Consisting of control lines and data lines
  - Control: determine which device gets to access the bus, what type of information on data lines
  - Data: address, command, data to and from devices
- Function:
  - Access control, arbitration: right to use bus
  - Ensure safe transaction in asynchronous transfer using hand-shaking protocol, and/or ECC

Buses

- Bunch of wires
  - Arbitration
  - Control/command
  - Data
  - Address
  - Flexible, low cost
  - Can be bandwidth bottleneck
- Types
  - Processor-memory
    - Short, fast, custom
  - I/O
    - Long, slow, standard
  - Backplane
    - Medium, medium, standard

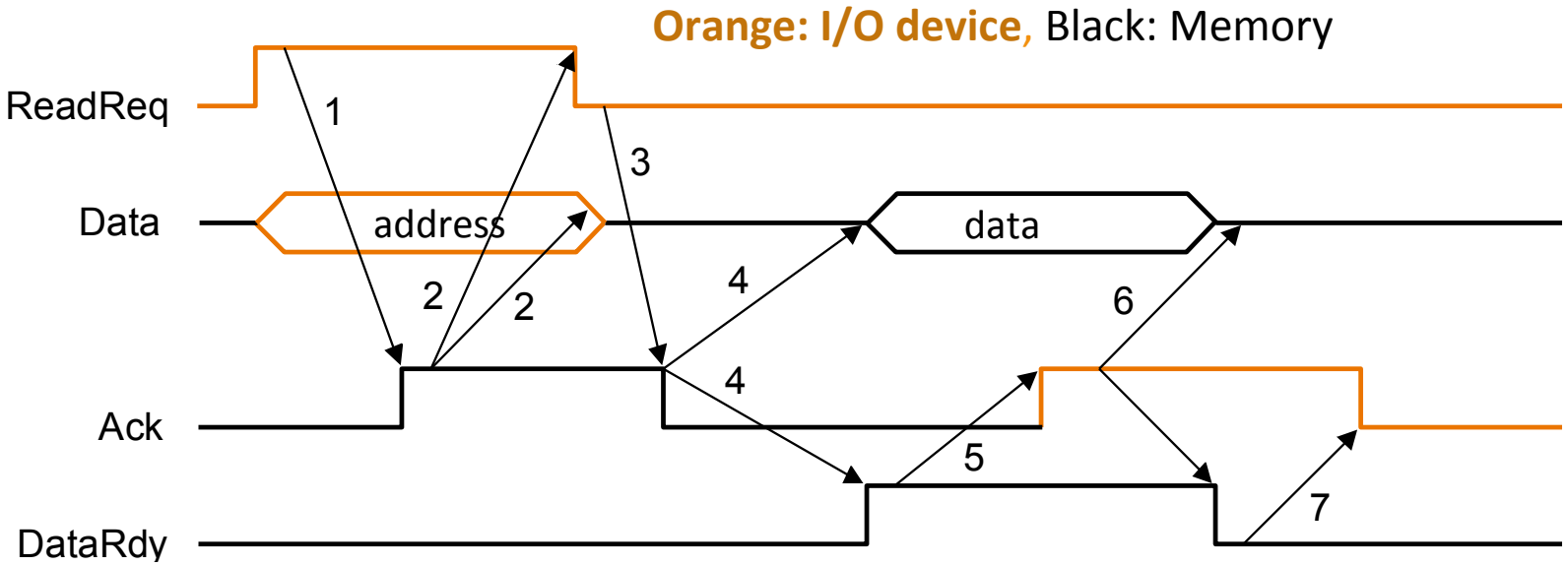
Buses in a Computer System



Buses

- Synchronous – has clock
  - Everyone watches clock and latches at appropriate phase
  - Transactions take fixed or variable number of clocks
  - Faster but clock limits length
  - E.g. processor-memory
- Asynchronous – requires handshake
  - More flexible
  - I/O
  - Handshaking protocol: A series of steps used to coordinate asynchronous bus transfers in which the sender and receiver proceed to the next step only when both parties agree that the current step has been completed.

Async. Handshake (Fig. 8.10)



(1) Request made by I/O device & (2) ack send by Memory

(3) Request deasserted & (4) ack deasserted

Wait till memory has data ready, it raise data ready and place data on data bus

(5) Data sent & (6) Data rec'd by I/O device and raise ack line & (7) ack deasserted and data off line by memory

Buses

- Improving bandwidth
  - Wider bus
  - Separate/multiplexed address/data lines
  - Block transfer
    - Spatial locality

Bus Arbitration

- Resolving bus control conflicts and assigning priorities to the requests for control of the bus.
- One or more bus masters, others slaves
  - Bus request
  - Bus grant
  - Priority
  - Fairness
- Implementations
  - Centralized (Arbiter, can be part of CPU or separate device)
  - Distributed (e.g. Ethernet)

Bus Mastering

- Allows bus to communicate directly with other devices on the bus without going through CPU
- Devices capable to take control of the bus.
- Frees up the processor (to do other work simultaneously)

Buses

- Bus architecture / standards
  - ISA (Industry Standard Architecture)
  - MCA (Micro Channel Architecture)
  - EISA (Extended Industry Standard Architecture)
  - VLB (Vesa Local Bus)
  - PCI (Peripheral Communications Interconnect)
- PCI
  - 32 or 64 bit
  - Synchronous 33MHz or 66MHz clock
  - Multiple masters
  - 111 MB/s peak bandwidth

Example: The Pentium 4's Buses

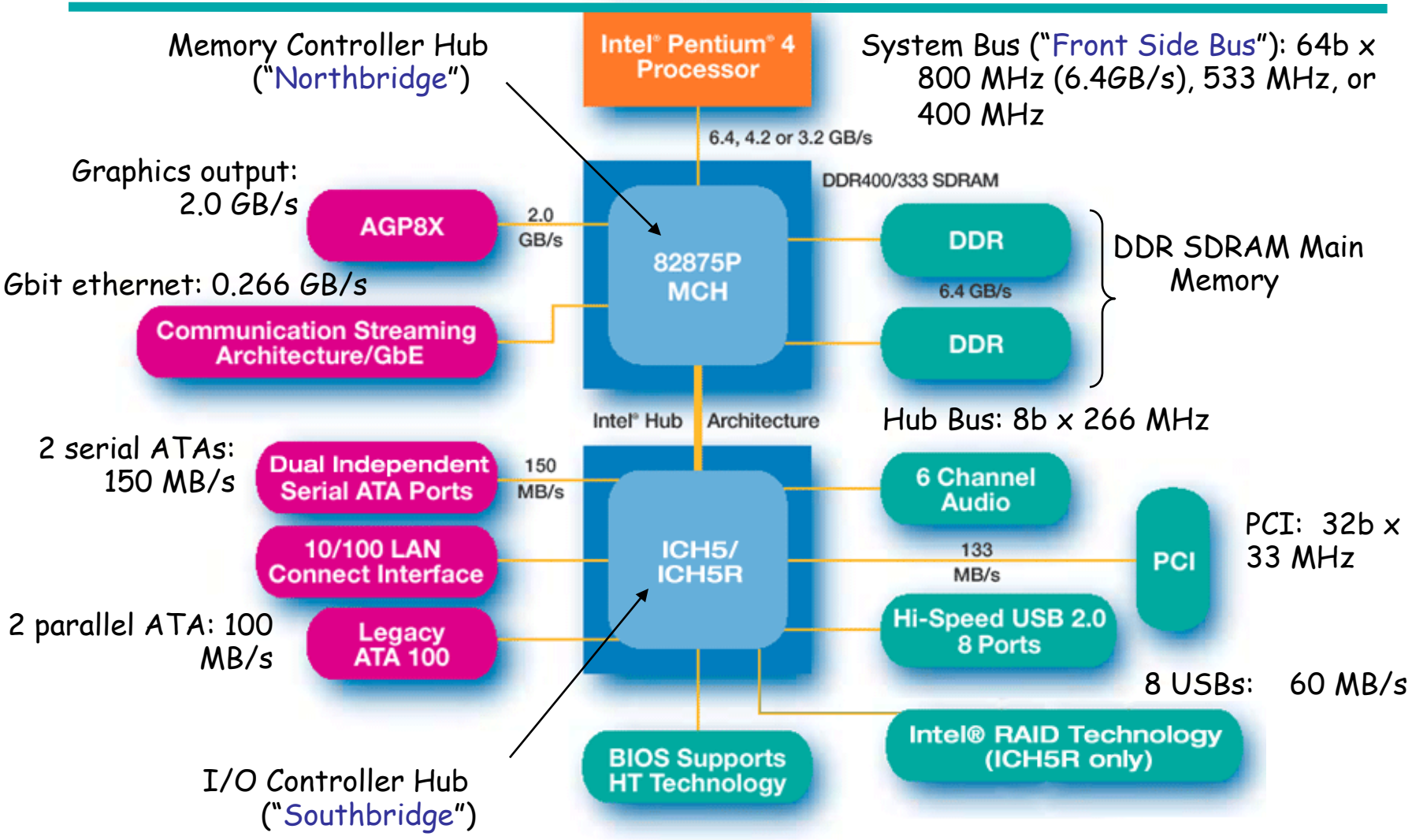
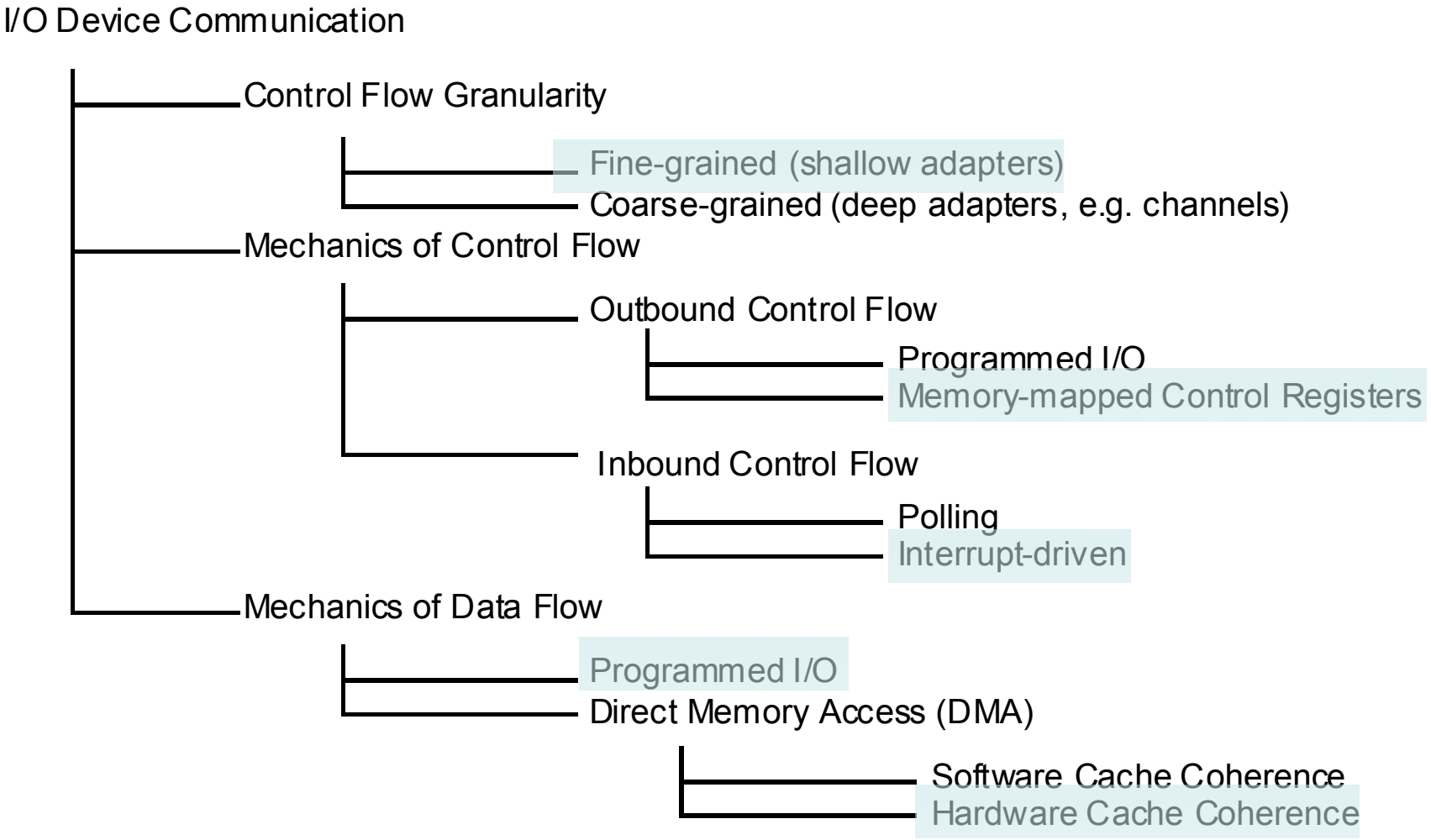


Fig 8.11

Interfacing

- Three key characteristics
  - Multiple users share I/O resource
  - Overhead of managing I/O can be high
  - Low-level details of I/O devices are complex
- Three key functions
  - Virtualize resources – protection, scheduling
  - Use interrupts (similar to exceptions)
  - Device drivers

Interfacing to I/O Devices



Interfacing

- How do you give I/O device a command?
  - Memory-mapped load/store
    - Special addresses not for memory
    - Send commands as data
    - Cacheable?
  - I/O commands
    - Special opcodes
    - Send over I/O bus

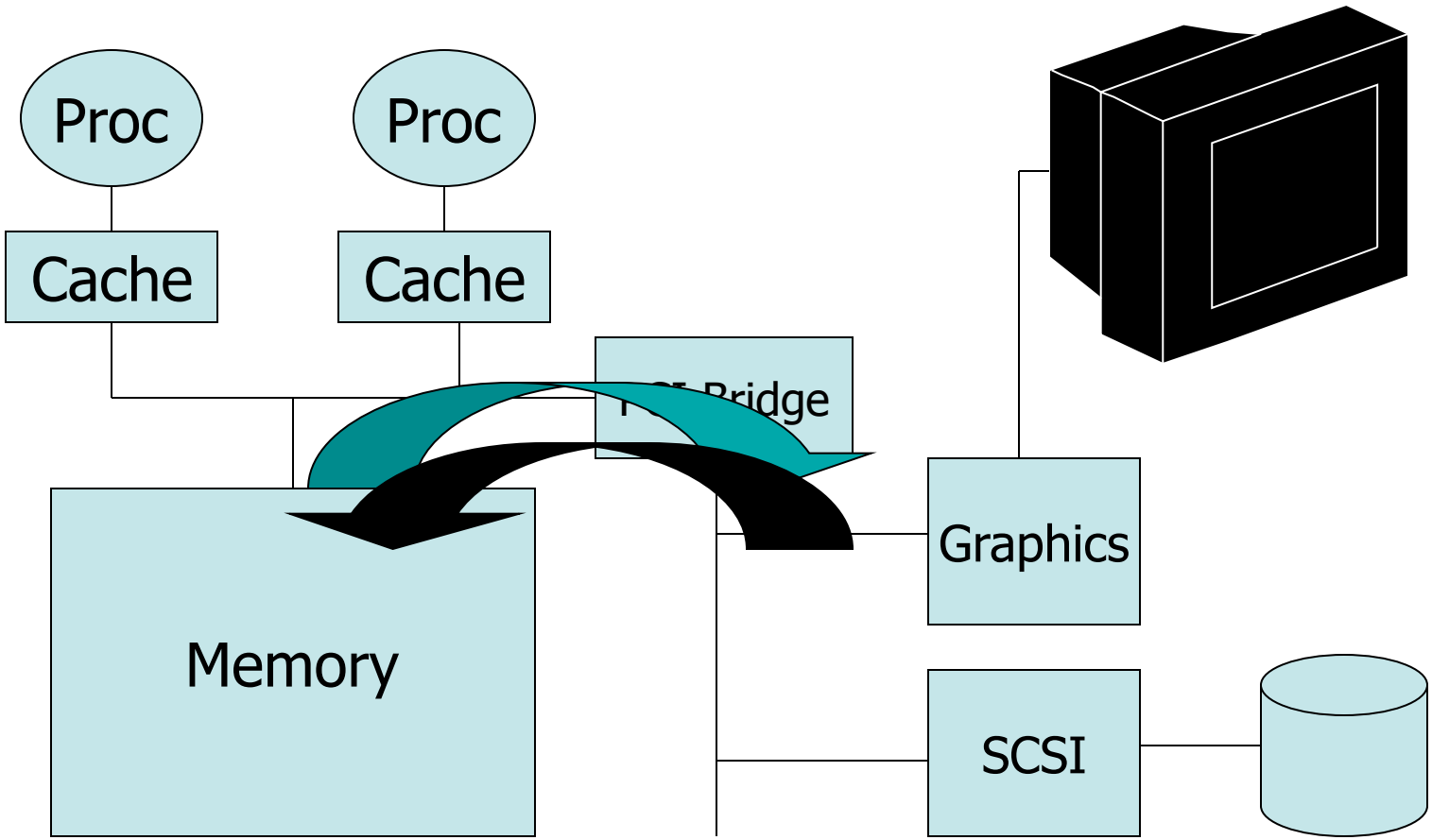
Interfacing

- How do I/O devices communicate w/ CPU?
  - Poll on devices
    - Waste CPU cycles
    - Poll only when device active?
  - Interrupts
    - Similar to exceptions, but asynchronous
    - Info in cause register
    - Possibly vectored interrupt handler

Interfacing

- Transfer data
  - Polling and interrupts – by CPU
  - OS transfers data
- Too many interrupts?
  - Use DMA so interrupt only when done
  - Use I/O channel – extra smart DMA engine
    - Offload I/O functions from CPU

Input/Output



Interfacing

- DMA
  - CPU sets up
    - Device ID, operation, memory address, # of bytes
  - DMA
    - Performs actual transfer (arb, buffers, etc.)
  - Interrupt CPU when done
- Typically I/O bus with devices use DMA
  - E.g. hard drive, NIC

Interfacing

- DMA virtual or physical addresses?
- Cross page boundaries within DMA?
  - Virtual
    - Page table entries, provided by OS
  - Physical
    - One page per transfer
    - OS chains the physical addresses
- No page faults in between – lock pages

Thank You