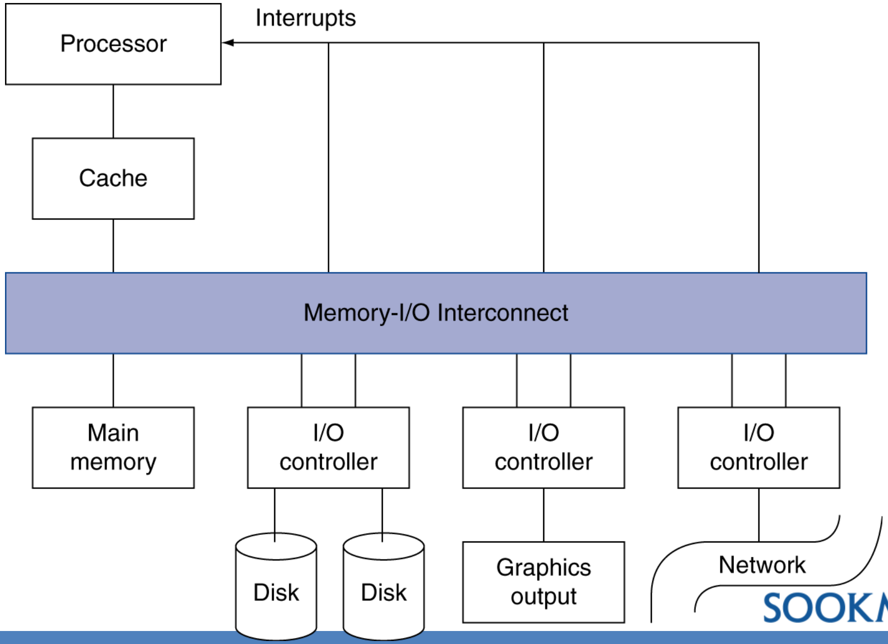
Storage and Other I/O Topics

Chulyun Kim

* This material is based on the lecture slides provided by Morgan Kaufmann

Introduction

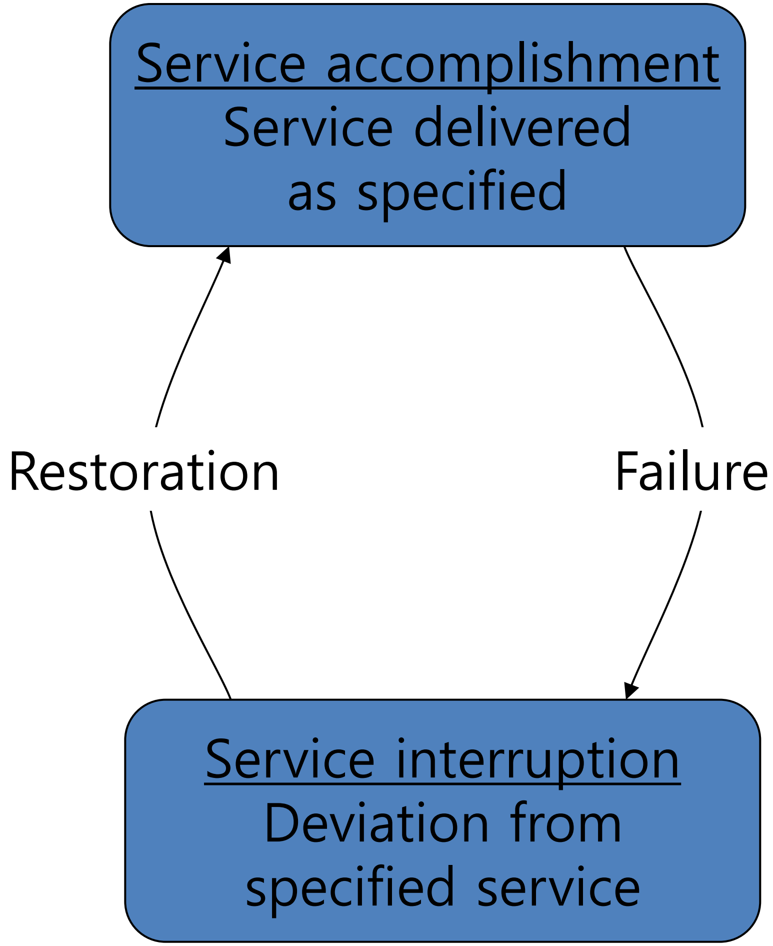
- I/O devices can be characterized by
  - Behavior: input, output, storage
  - Partner: human or machine
  - Data rate: bytes/sec
- I/O bus connections



I/O System Characteristics

- Dependability is important
  - Particularly for storage devices
- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)
  - Desktops & embedded systems
    - Mainly interested in response time & diversity of devices
  - Servers
    - Mainly interested in throughput & expandability of devices

Dependability



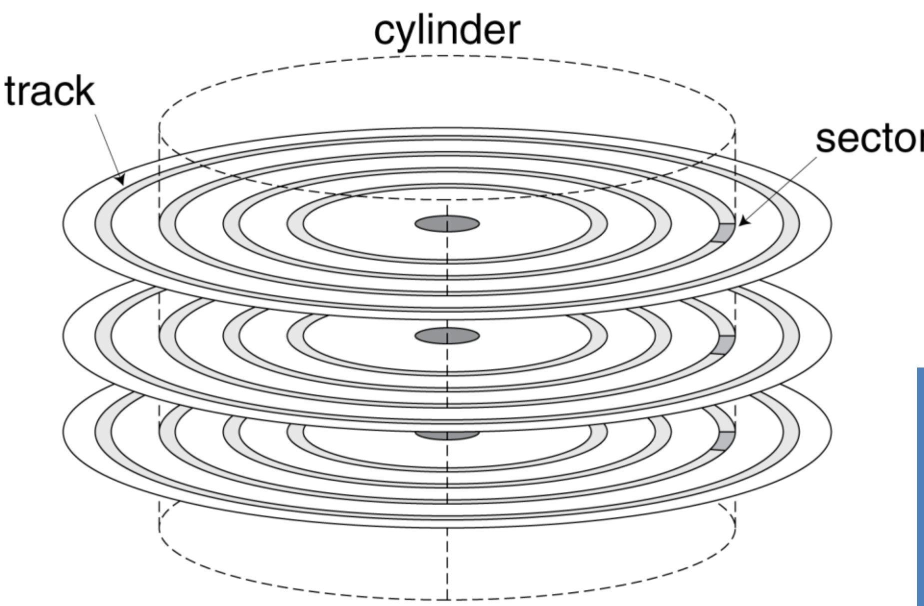
- Fault: failure of a component
  - May or may not lead to system failure

Dependability Measures

- Reliability: mean time to failure (MTTF)
- Service interruption: mean time to repair (MTTR)
- Mean time between failures
  - MTBF = MTTF + MTTR
- Availability = MTTF / (MTTF + MTTR)
- Improving Availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis and repair

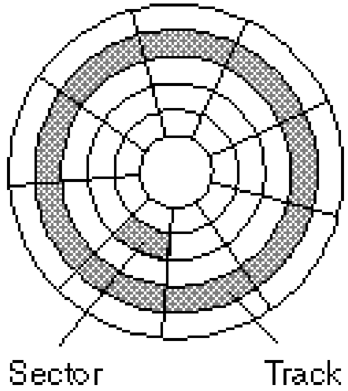
Disk Storage

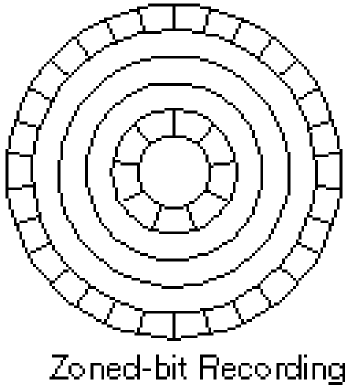
- Nonvolatile, rotating magnetic storage



Disk Sectors and Access

- Each sector records
  - Zone bit recording (ZBR)
  - Data (512 bytes, 4096 bytes proposed)
- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead





Disk Access Example

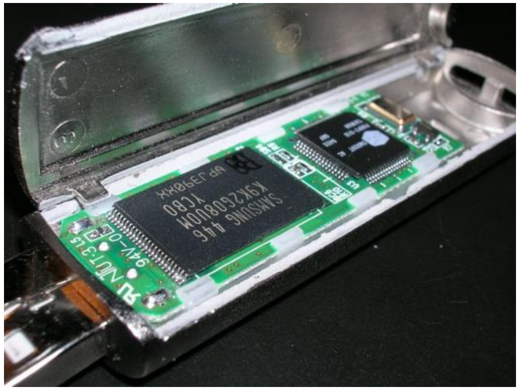
- Given
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- Average read time
  - 4ms seek time
    + ½ / (15,000/60) = 2ms rotational latency
    + 512 / 100MB/s = 0.005ms transfer time
    + 0.2ms controller delay
    = 6.2ms
- If actual average seek time is 1ms
  - Average read time = 3.2ms

Disk Performance Issues

- Manufacturers quote average seek time
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times
- Smart disk controller allocate physical sectors on disk
  - Present logical sector interface to host
  - SCSI, ATA, SATA
- Disk drives include caches
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay

Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)



Flash Types

- NOR flash: bit cell like a NOR gate
  - Random read/write access
  - Used for instruction memory in embedded systems
- NAND flash: bit cell like a NAND gate
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, …
- Flash bits wears out after 1000's of accesses
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks
- Asynchronous speed between read and write

Interconnecting Components

- Need interconnections between
  - CPU, memory, I/O controllers
- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer
  - Can become a bottleneck
- Performance limited by physical factors
  - Wire length, number of connections
- More recent alternative: high-speed serial connections with switches
  - Like networks

Bus Types

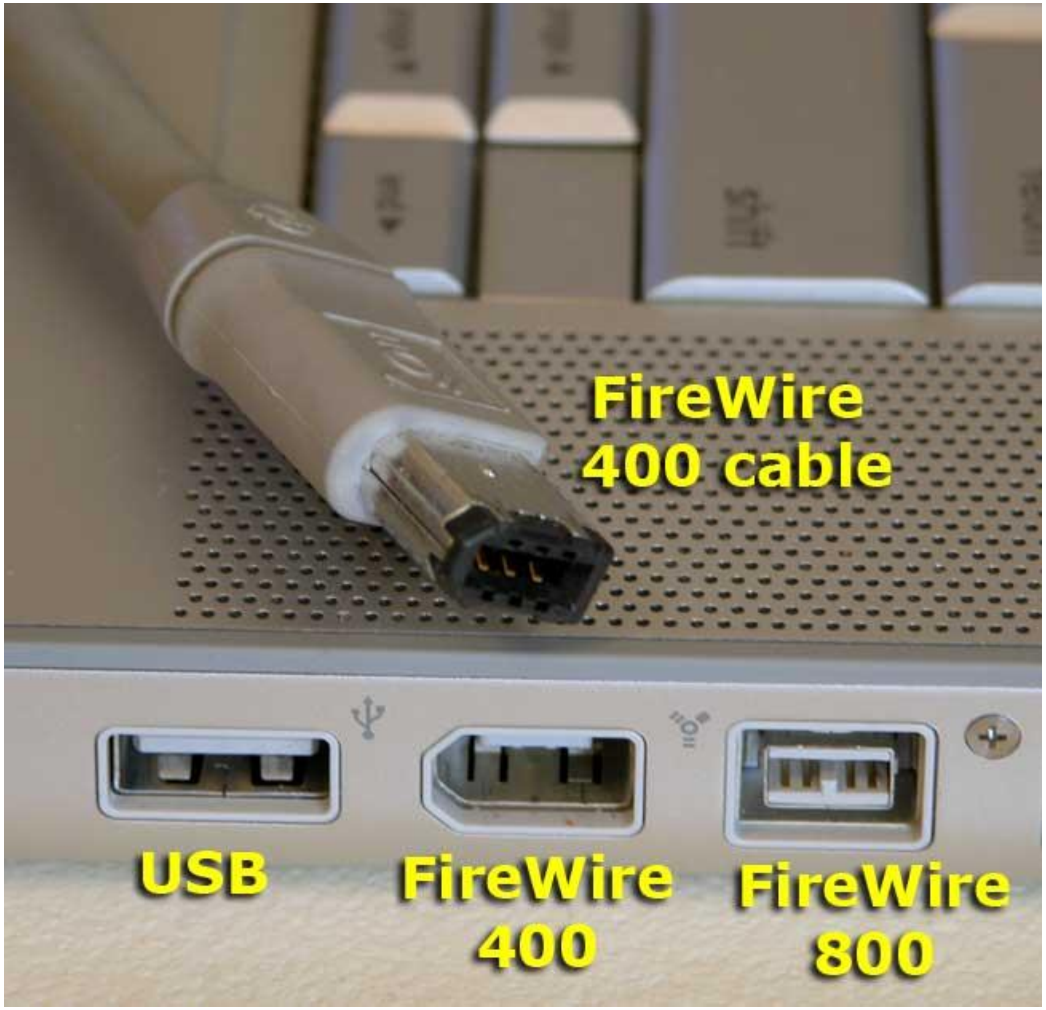
- Processor-Memory buses
  – Short, high speed
  – Design is matched to memory organization
- I/O buses
  – Longer, allowing multiple connections
  – Specified by standards for interoperability
  – Connect to processor-memory bus through a bridge

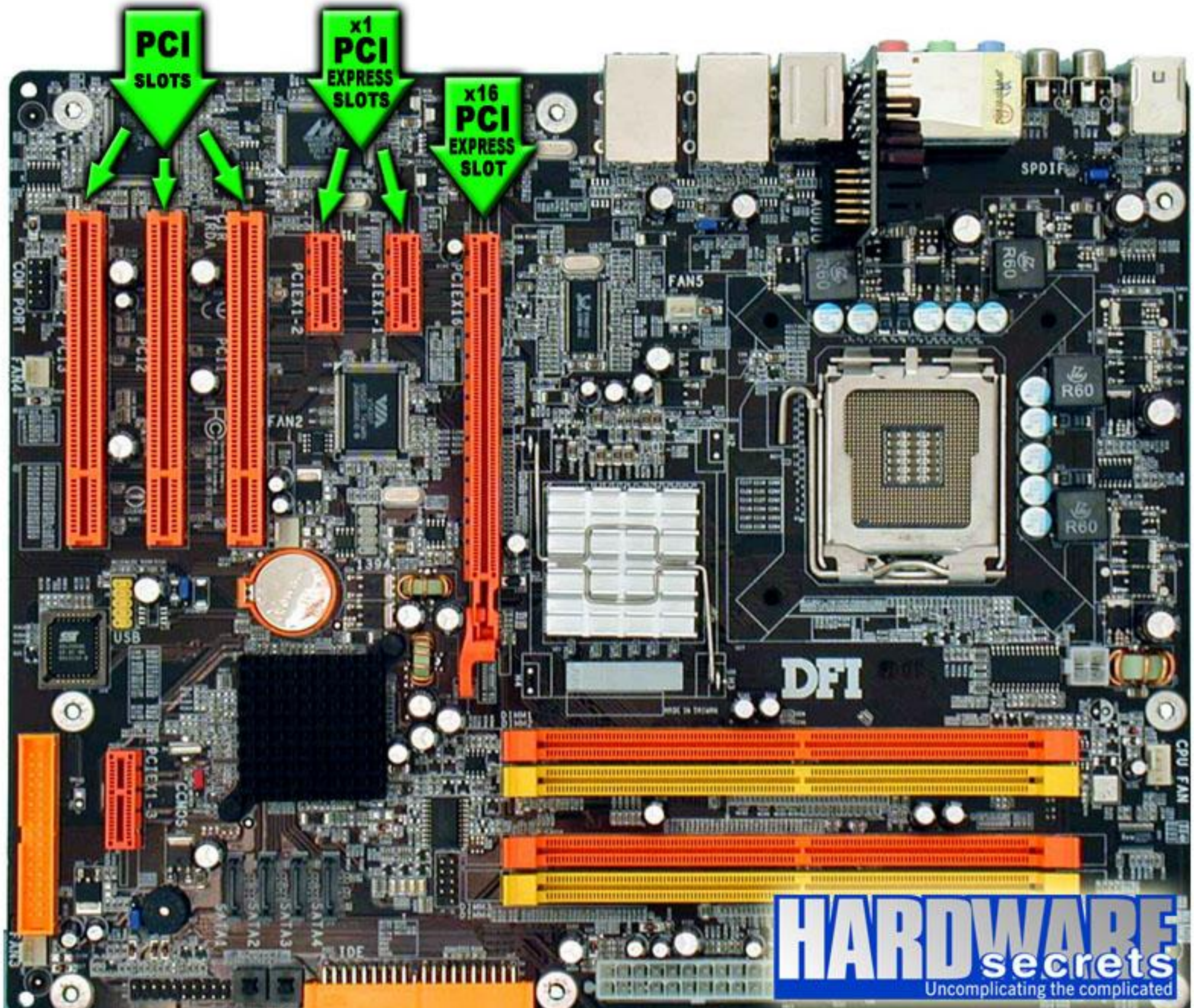
Bus Signals and Synchronization

- Data lines
  - Carry address and data
  - Multiplexed or separate
- Control lines
  - Indicate data type, synchronize transactions
- Synchronous
  - Uses a bus clock
- Asynchronous
  - Uses request/acknowledge control lines for handshaking

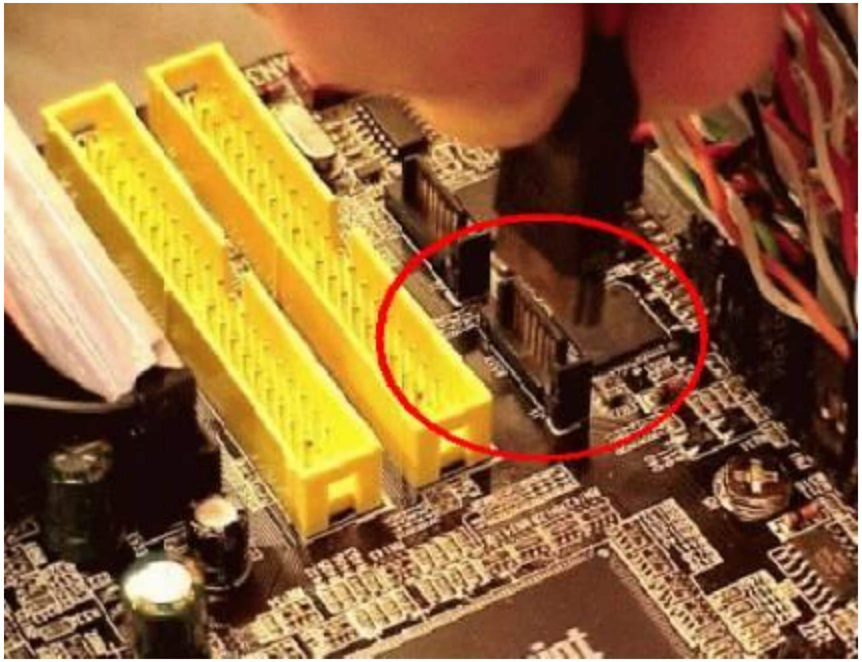
I/O Bus Examples

	Firewire	USB 2.0	PCI Express	Serial ATA	Serial Attached SCSI
Intended use	External	External	Internal	Internal	External
Devices per channel	63	127	1	1	4
Data width	4	2	2/lane	4	4
Peak bandwidth	50MB/s or 100MB/s	0.2MB/s, 1.5MB/s, or 60MB/s	250MB/s/lane 1×, 2×, 4×, 8×, 16×, 32×	300MB/s	300MB/s
Hot pluggable	Yes	Yes	Depends	Yes	Yes
Max length	4.5m	5m	0.5m	1m	8m
Standard	IEEE 1394	USB Implementers Forum	PCI-SIG	SATA-IO	INCITS TC T10

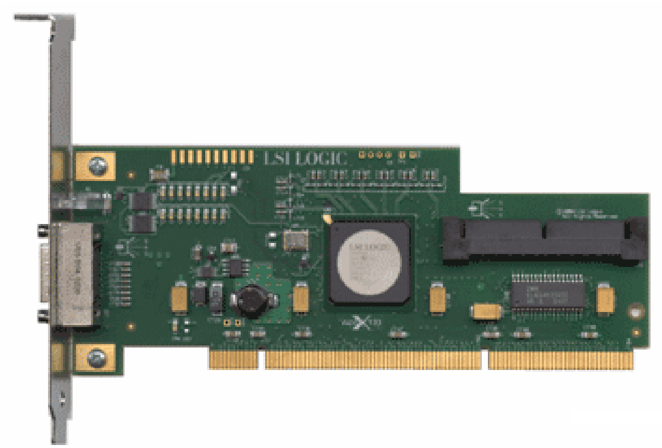

FireWire
400 cable
USB
FireWire
400
FireWire
800


PCI SLOTS
x1 PCI EXPRESS SLOTS
x16 PCI EXPRESS SLOT
SPDIF
FAN5
FAN2
FAN4
COM PORT
USB
1394
CCMOS
SATA1
SATA2
SATA3
SATA4
IDE
CPU FAN
R60
DFI
HARDWARE secrets
Uncomplicating the complicated

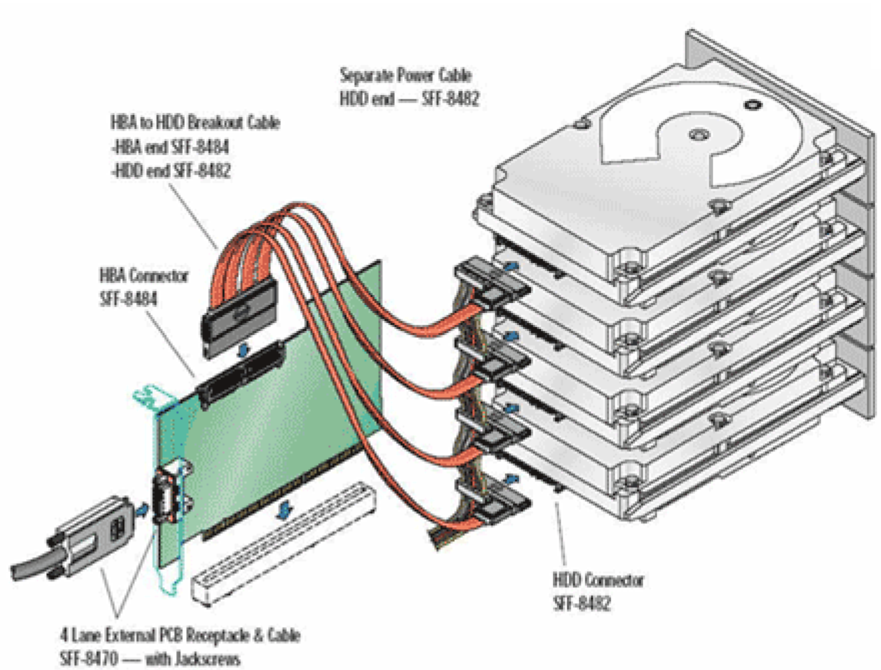
VERSITY



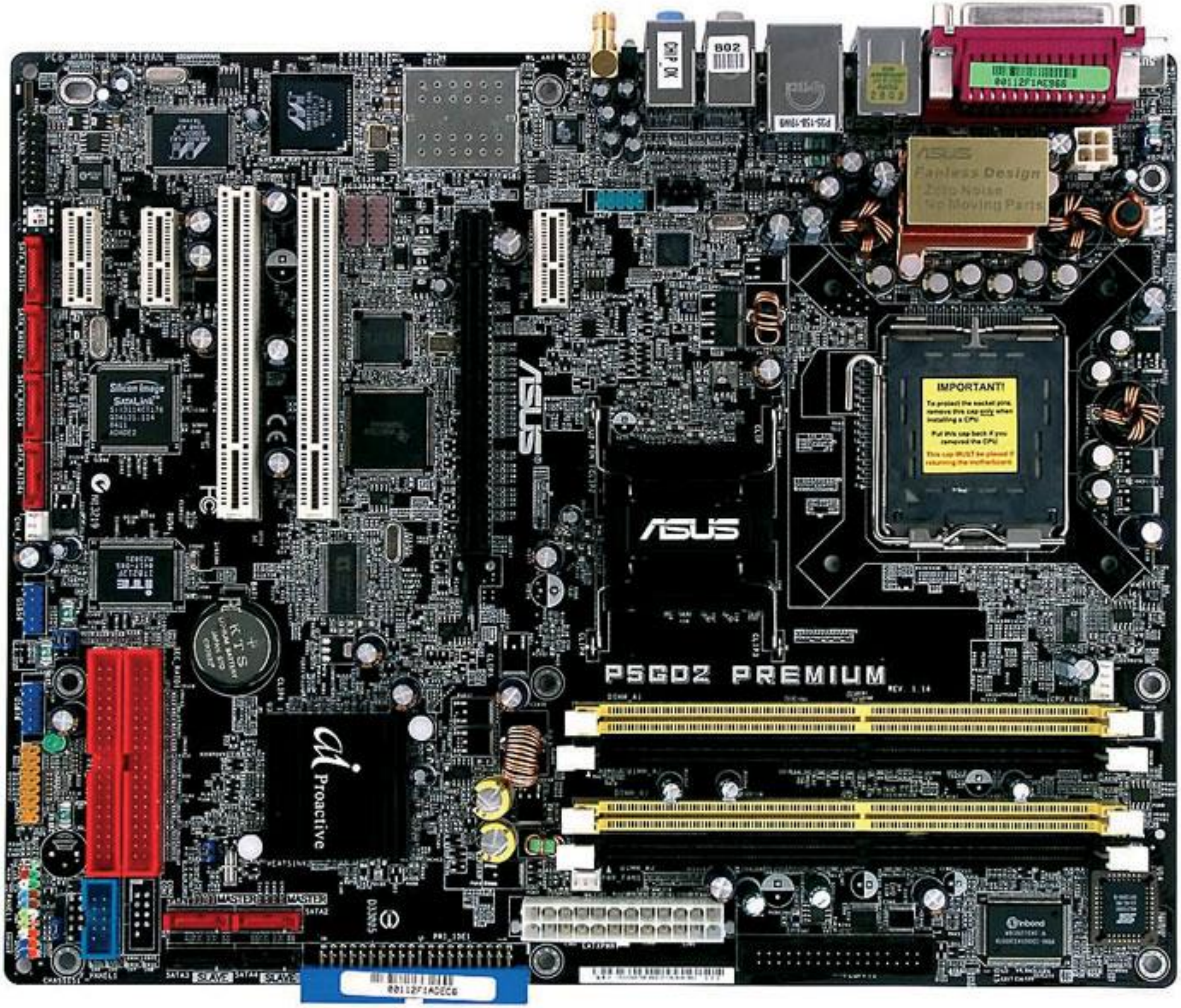

Serial ATA

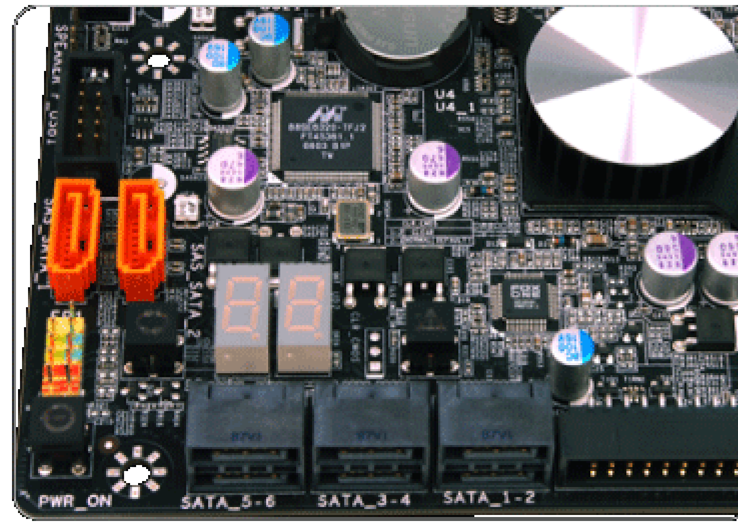


LSI LOGIC

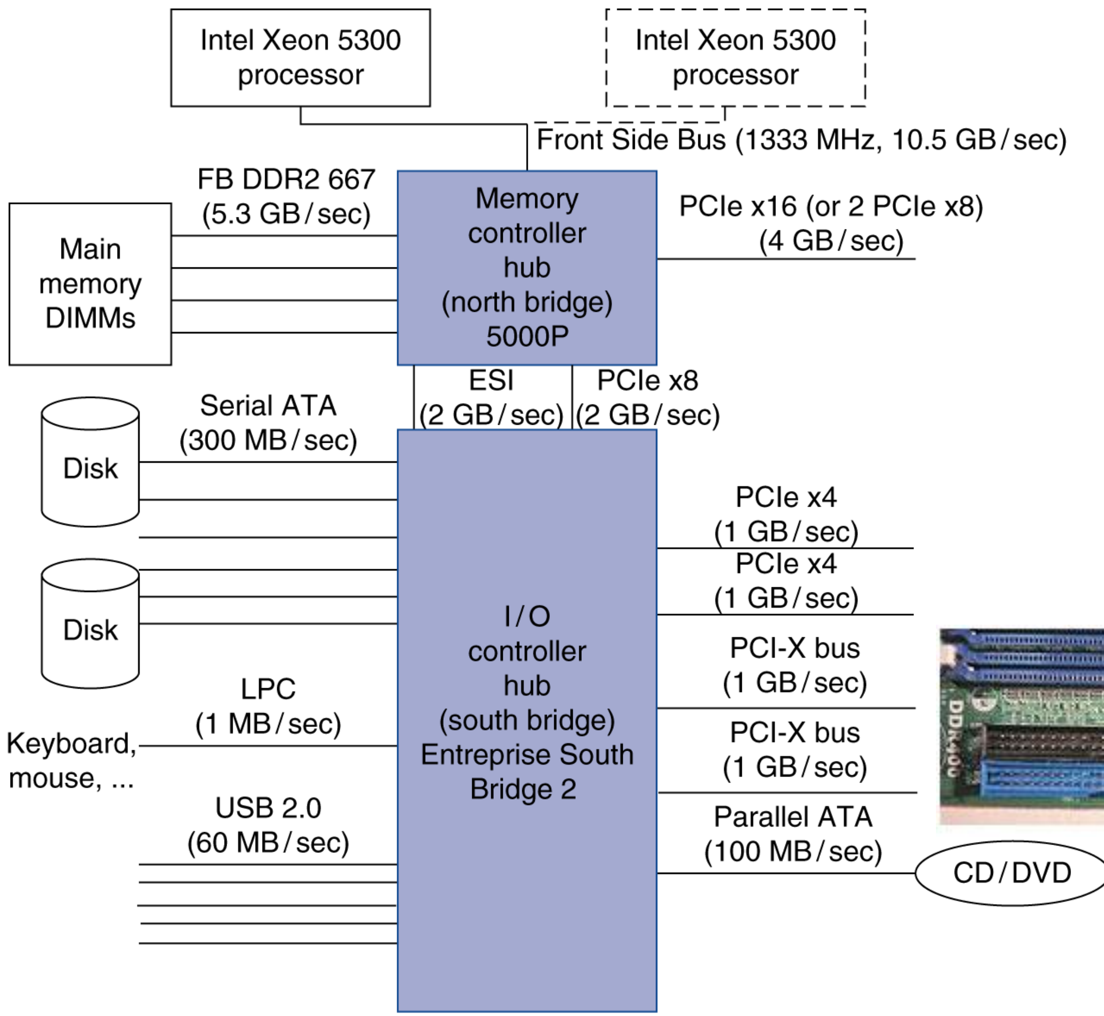


Separate Power Cable
HDD end — SFF-8482
HBA to HDD Breakout Cable
-HBA end SFF-8484
-HDD end SFF-8482
HBA Connector
SFF-8484
HDD Connector
SFF-8482
4 Lane External PCB Receptacle & Cable
SFF-8470 — with Jackscrews


ASUS
P5GD2 PREMIUM
IMPORTANT!
Fanless Design
Proactive


PWR_ON
SATA_5-6
SATA_3-4
SATA_1-2

Typical x86 PC I/O System



I/O Management

- I/O is mediated by the OS
  - Multiple programs share I/O resources
    - Need protection and scheduling
  - I/O causes asynchronous interrupts
    - Same mechanism as exceptions
  - I/O programming is fiddly
    - OS provides abstractions to programs

I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
  - Synchronizes operations with software
- Command registers
  - Cause device to do something
- Status registers
  - Indicate what the device is doing and occurrence of errors
- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device

I/O Register Mapping

- Memory mapped I/O
  - Registers are addressed in same space as memory
  - Address decoder distinguishes between them
- I/O instructions
  - Separate instructions to access I/O registers

Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action
- Common in small or low-performance real-time embedded systems
  - Predictable timing
  - Low hardware cost
- In other systems, wastes CPU time

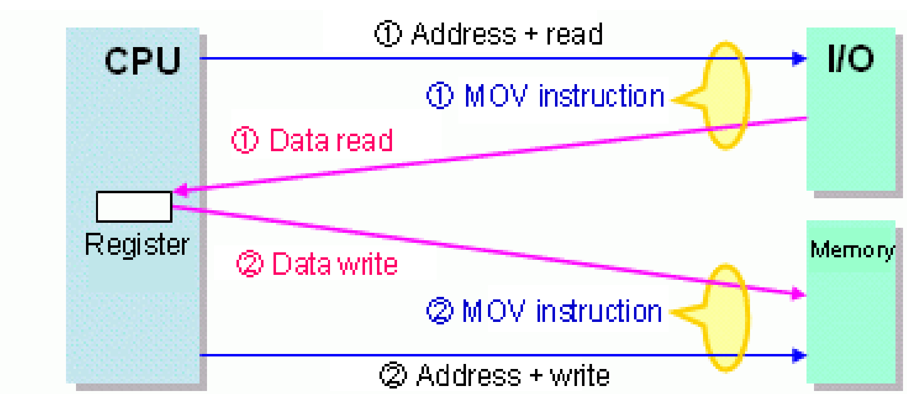
Interrupts

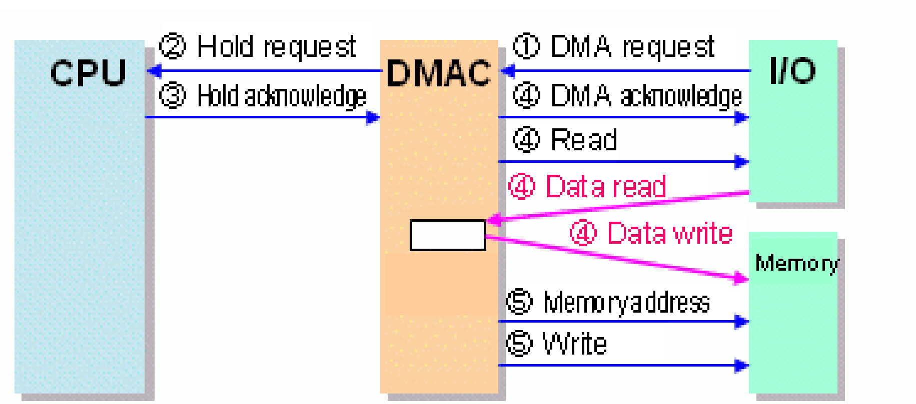
- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device
- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt

I/O Data Transfer

- Polling and interrupt-driven I/O
  - CPU transfers data between memory and I/O data registers
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error

DMA


CPU
① Address + read
I/O
① MOV instruction
① Data read
Register
② Data write
Memory
② MOV instruction
② Address + write


CPU
② Hold request
③ Hold acknowledge
DMAC
① DMA request
④ DMA acknowledge
I/O
④ Read
④ Data read
④ Data write
Memory
⑤ Memory address
⑤ Write

I/O vs. CPU Performance

- Amdahl's Law
  - Don't neglect I/O performance as parallelism increases compute performance
- Example
  - Benchmark takes 90s CPU time, 10s I/O time
  - Double the number of CPUs/2 years
    - I/O unchanged

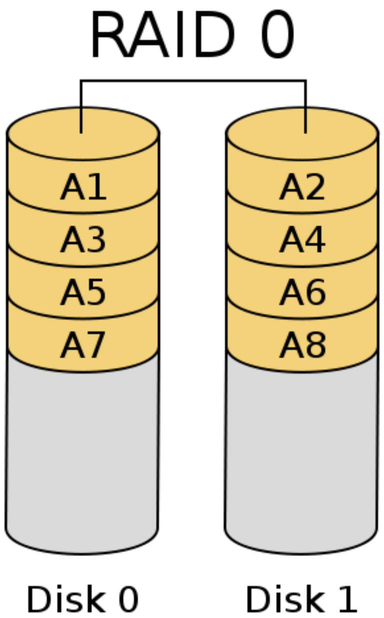
Year	CPU time	I/O time	Elapsed time	% I/O time
now	90s	10s	100s	10%
+2	45s	10s	55s	18%
+4	23s	10s	33s	31%
+6	11s	10s	21s	47%

RAID

- Redundant Array of Inexpensive (Independent) Disks
  - Use multiple smaller disks (c.f. one large disk)
  - Parallelism improves performance
  - Plus extra disk(s) for redundant data storage
- Provides fault tolerant storage system
  - Especially if failed disks can be “hot swapped”

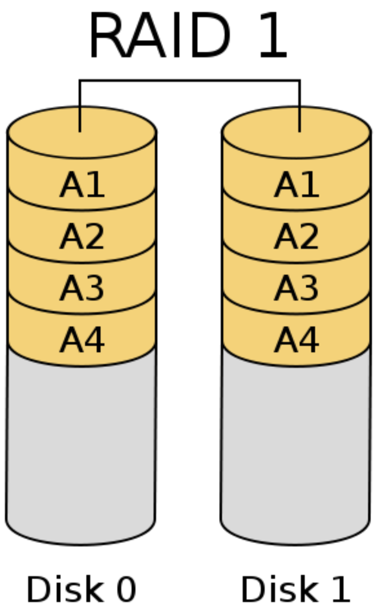
RAID 0

- N+0 disks
- Block-level striping without parity or mirroring
  - Just stripe data over multiple disks
- But it does improve performance



RAID 1

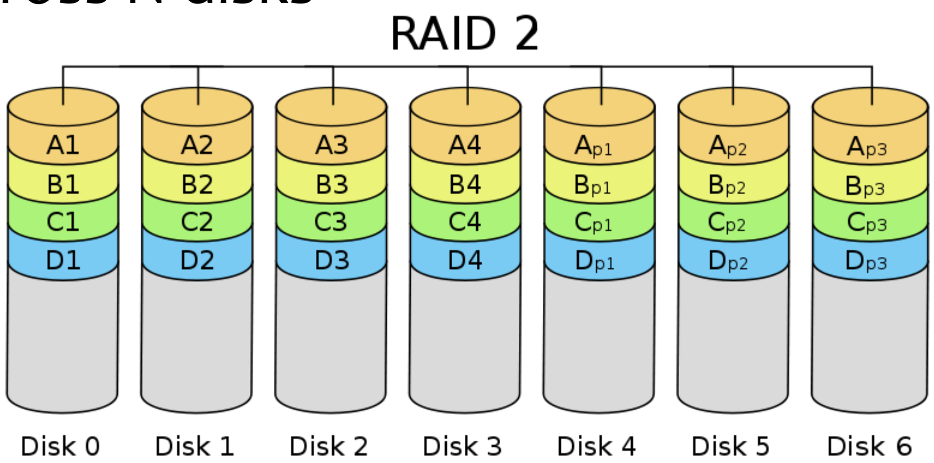
- N+N disks
- Mirroring without parity or striping
  - Write data to both data disk and mirror disk
  - On disk failure, read from mirror



RAID 2

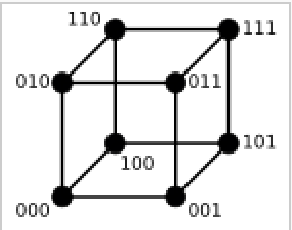
– N + E disks (e.g., 10 + 4)

- Bit-level striping with dedicated Hamming-code parity
- Error correcting code (ECC)
  - Split data at bit level across N disks
  - Generate E-bit ECC
  - Not used in practice
- Space Efficiency: N
- Fault Tolerance: E

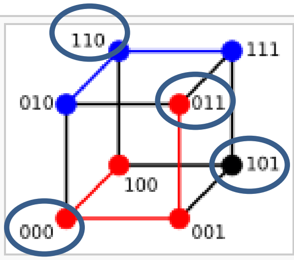


Error Correcting Code

- Hamming Distance
  - The Hamming distance is the number of bits that have to be changed to get from one bit pattern to another
  - Example: 10010101 & 10011001 have a hamming distance of 2
  - For any coding whose members have a Hamming distance of two, any one bit error can be detected



3-bit binary cube for finding hamming distance



Two example distances: 100->011 has distance 3 (red path); 010->111 has distance 2 (blue path)

Error Correcting Code

- If we compare the read K bits compared with the write K bits, using an EXOR function, the result is called the "syndrome"
- If the syndrome is all zeros, there were no errors
- If there is a 1 bit somewhere, we know it represents an error

Error Correcting Code

- Hamming Code Design
  - To store an M bit word with detection/correction takes M+K bit words
  - If K =1, we can detect single bit errors but not correct them
  - If $2^K - 1 >= M + K$, we can detect, identify, and correct all single bit errors, i.e. the syndrome contains the information to correct any single bit error

Example: For M = 8:

and K = 3: $2^3 - 1 = 7 < 8 + 3$ (doesn't work)

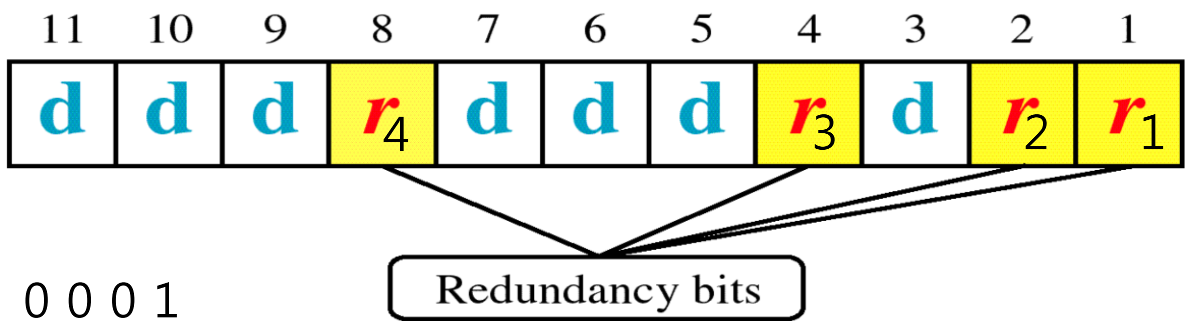
and K = 4: $2^4 - 1 = 15 > 8 + 4$ (works!)

Therefore, we must choose K =4,

i.e., the minimum size of the syndrome is 4

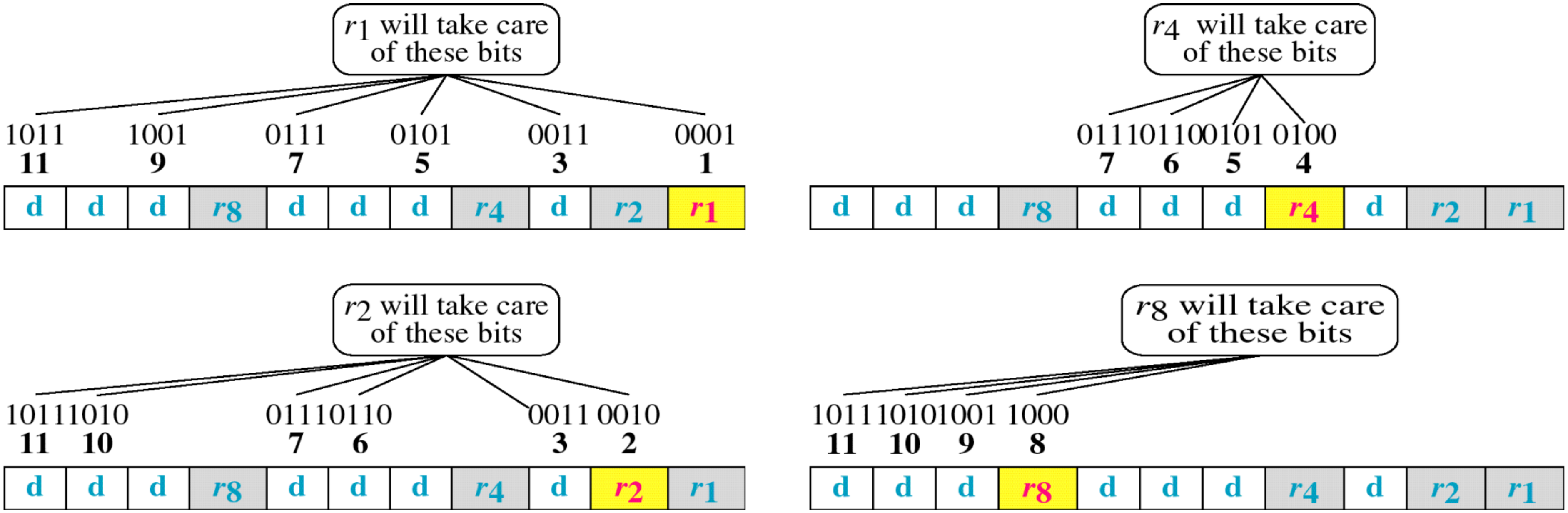
Error Correcting Code

- Example

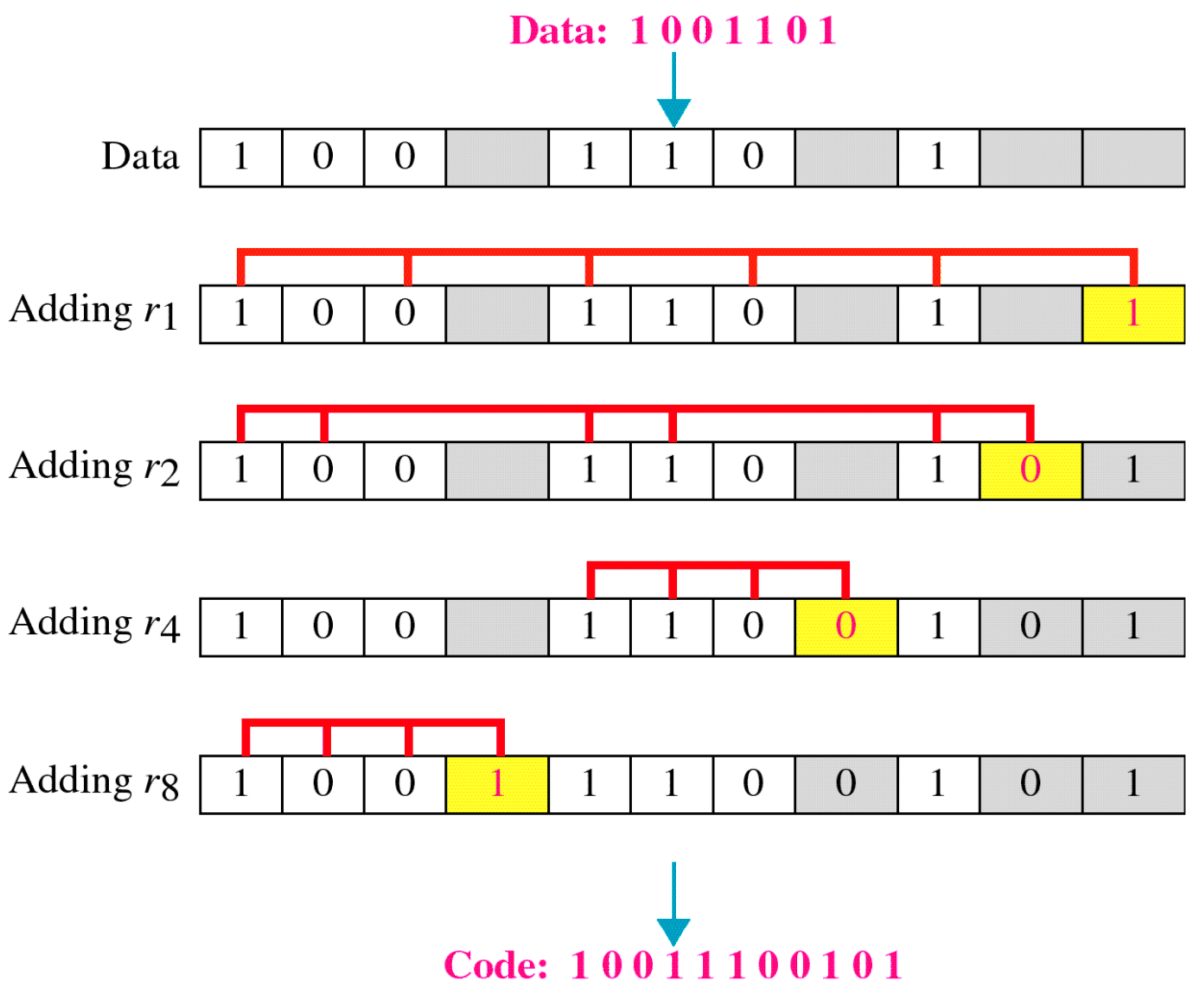


1:	0 0 0 1
2:	0 0 1 0
3:	0 0 1 1
4:	0 1 0 0
5:	0 1 0 1
6:	0 1 1 0
7:	0 1 1 1
8:	1 0 0 0
9:	1 0 0 1
10:	1 0 1 0
11:	1 0 1 1

Error Correcting Code

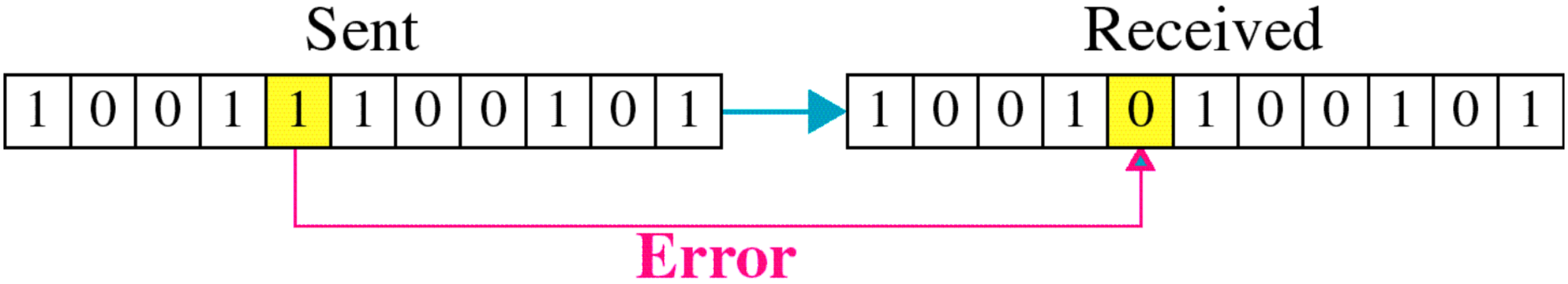


Error Correcting Code



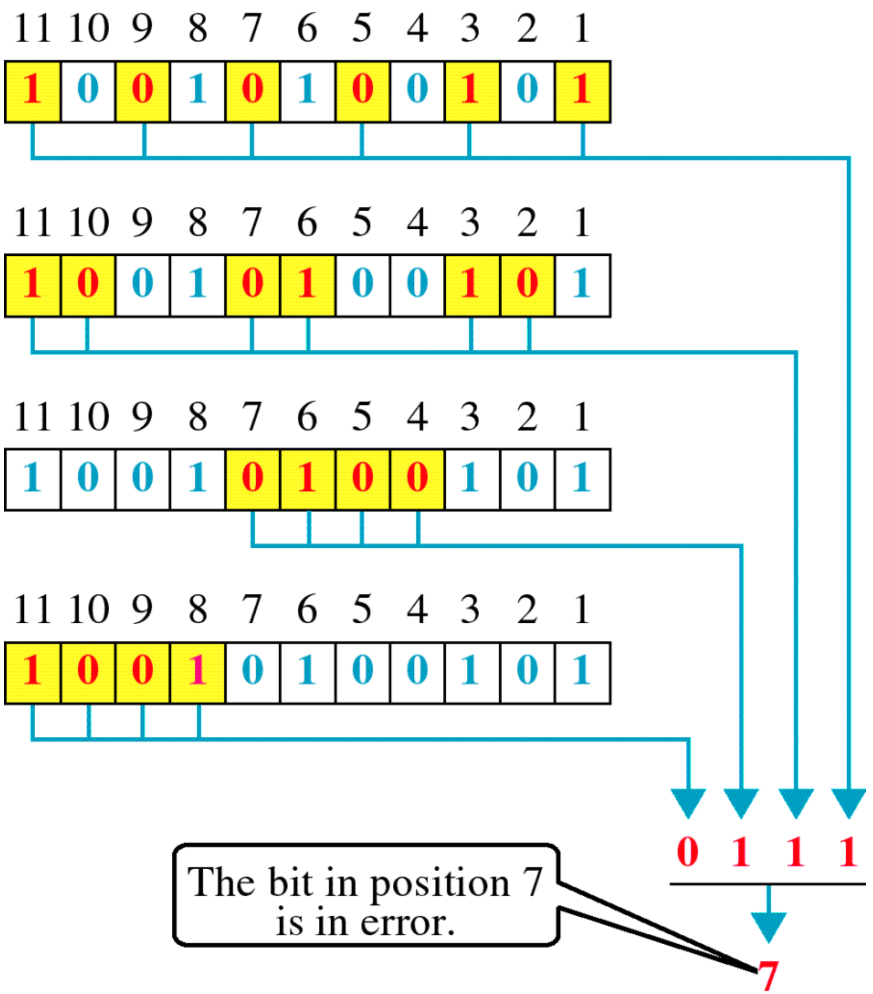
Error Correcting Code

- Single-bit error



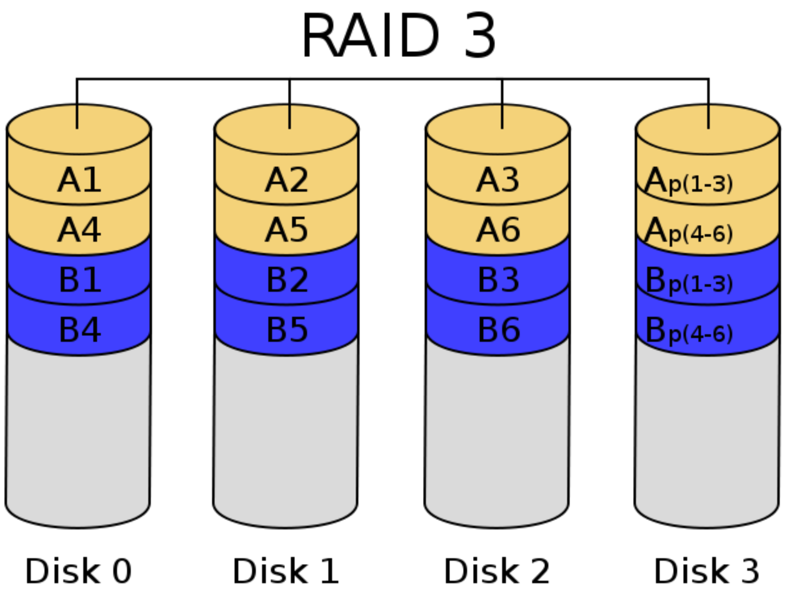
Error Correcting Code

- Error Correction



RAID 3

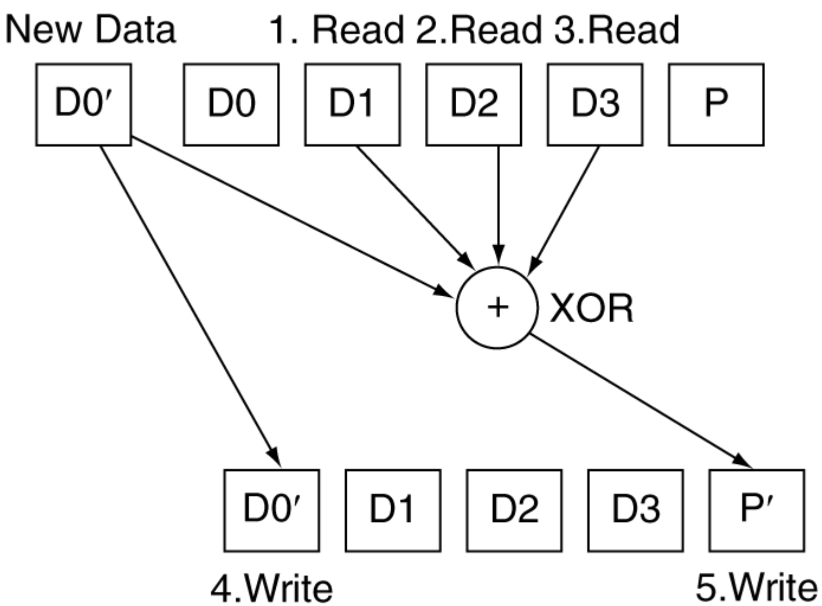
- N + 1 disks
- Byte-level striping with dedicated parity
  - Data striped across N disks at byte level
  - Redundant disk stores parity
  - Read access
    - Read all disks
  - Write access
    - Generate new parity
    - update all disks
  - On failure
    - Use parity to reconstruct
- Not widely used



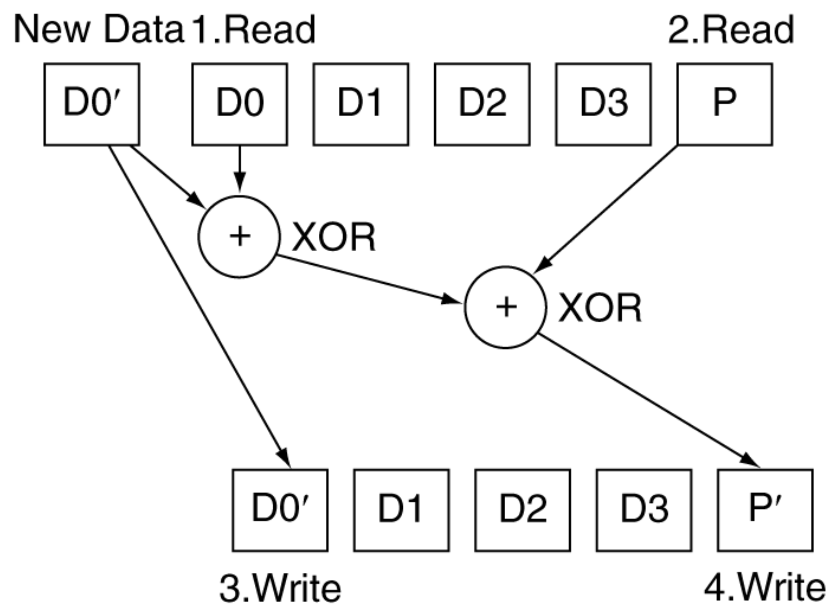
RAID 4

- N + 1 disks
- Block-level striping with dedicated parity
  - Data striped across N disks at block level
  - Redundant disk stores parity for a group of blocks
  - Read access
    - Read only the disk holding the required block
  - Write access
    - Just read disk containing modified block, and parity disk
    - Calculate new parity, update data disk and parity disk
  - On failure
    - Use parity to reconstruct
- Not widely used

RAID 3 vs RAID 4



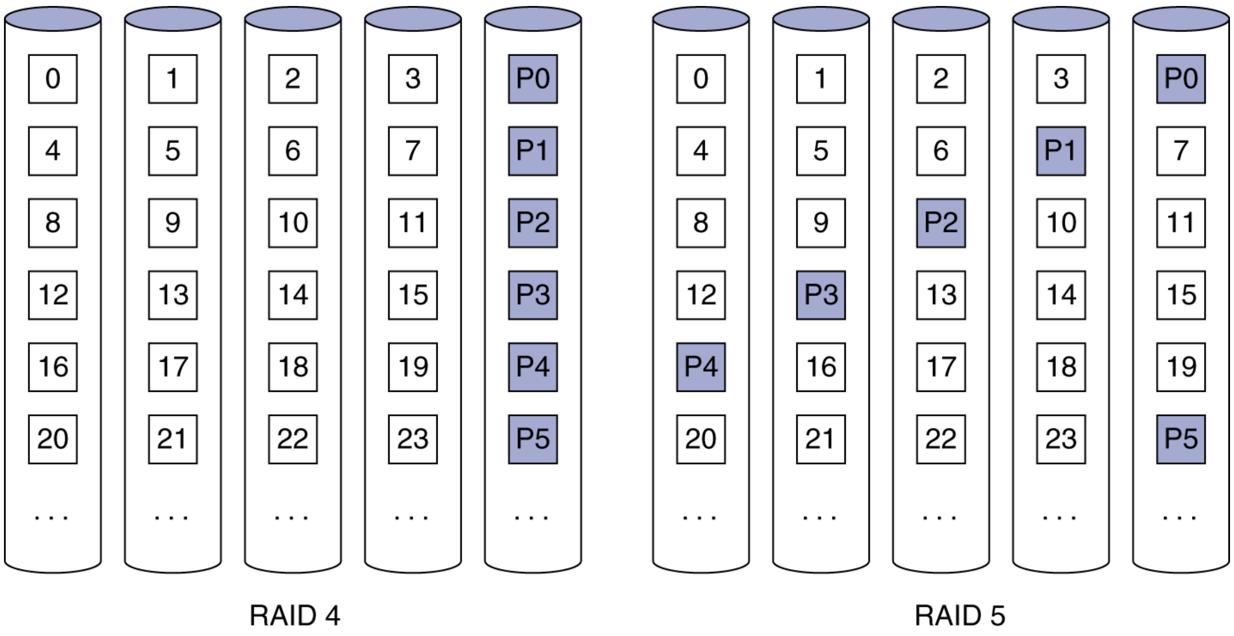
Bit-Interleaved Parity



Block-Interleaved Parity

RAID 5: Distributed Parity

- N + 1 disks
  - Like RAID 4, but parity blocks distributed across disks
    - Avoids parity disk being a bottleneck
- Widely used



RAID Summary

- RAID can improve performance and availability
  - High availability requires hot swapping
- Assumes independent disk failures
  - Too bad if the building burns down!
- See "Hard Disk Performance, Quality and Reliability"
  - http://www.pcguide.com/ref/hdd/perf/index.htm

Concluding Remarks

- I/O performance measures
  - Throughput, response time
  - Dependability and cost also important
- Buses used to connect CPU, memory, I/O controllers
  - Polling, interrupts, DMA
- RAID
  - Improves performance and dependability