Chapter 6B

Storage and Other I/O Topics

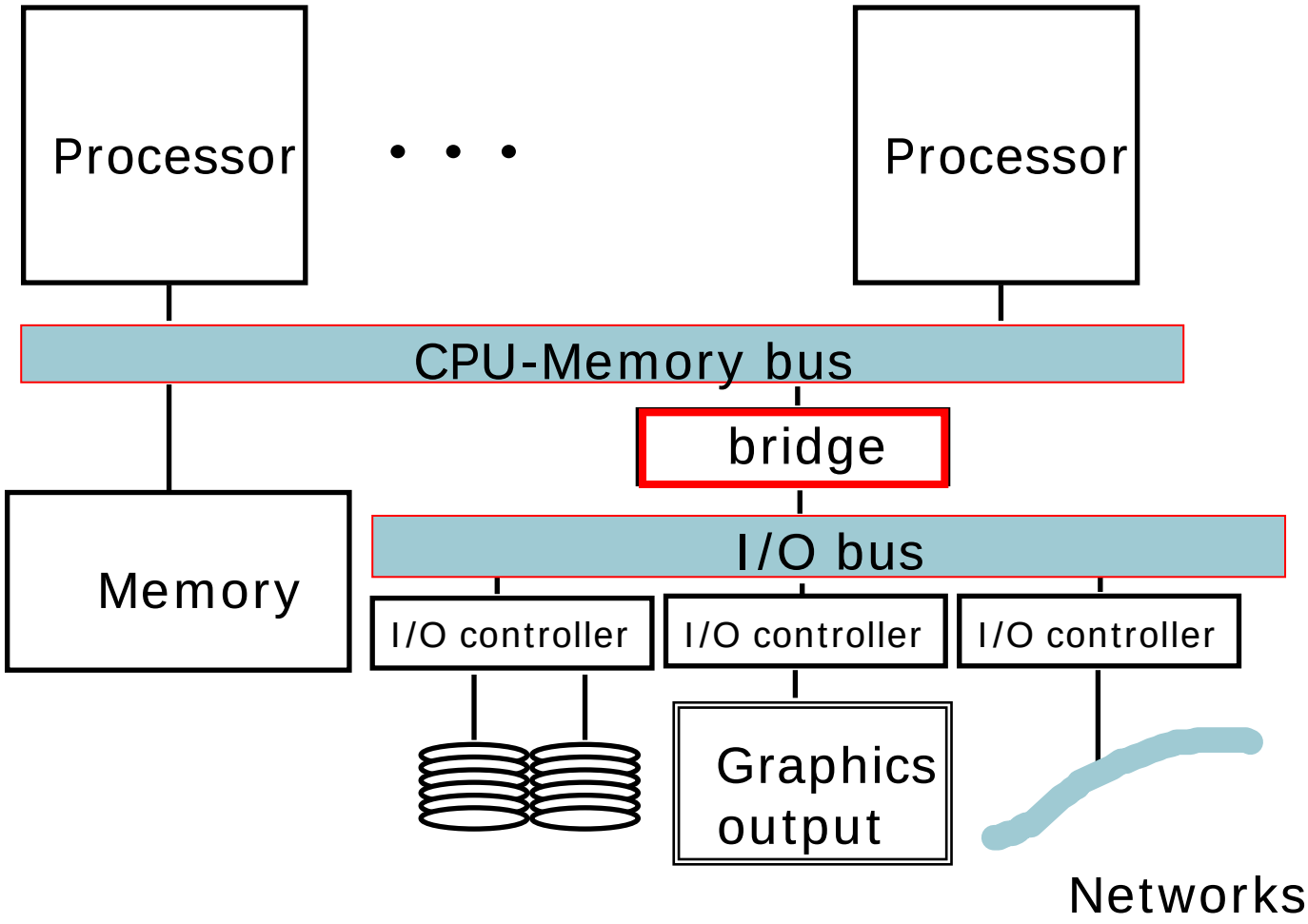
Interconnecting Components

- 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
- Backplane bus
  - A bus that is designed to allow processor, memory, and I/O devices to coexist on a single bus.

Bridge


Processor
Processor
CPU-Memory bus
bridge
Memory
I/O bus
I/O controller
I/O controller
I/O controller
Graphics output
Networks

Connection Basics

- An I/O transaction includes two parts: sending the address and sending or receiving the data
  - A sequence of operations over the interconnect that includes a request and may include a response
  - A read transaction or a write transaction
- Several standards exist
  - Firewire, USB, PCI express (PCIe), Serial ATA (SATA), Serial attached SCSI (SAS)

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

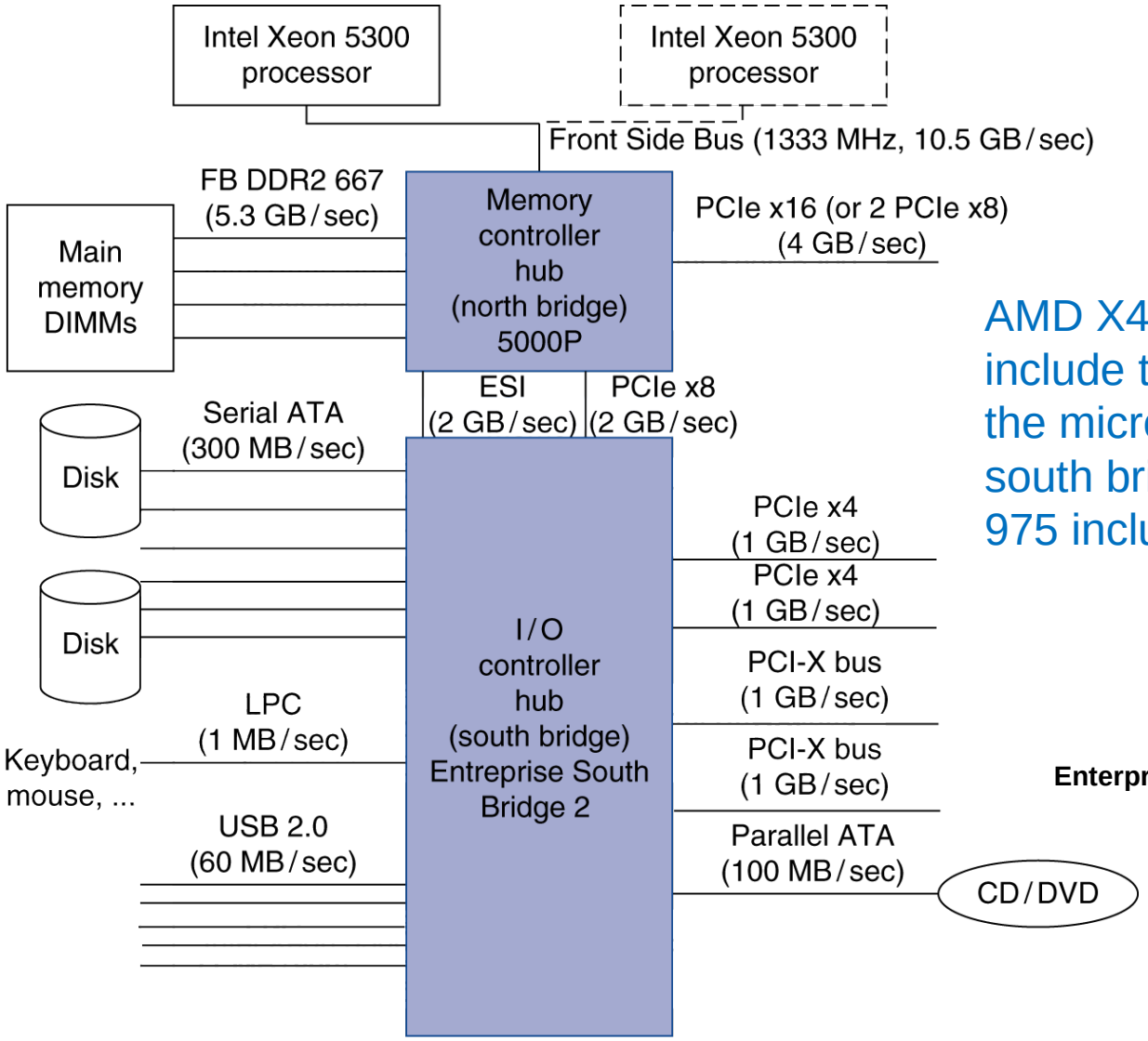
Bus Signals and Synchronization

- 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

Handshaking Protocols

- Used in an asynchronous bus
- 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.

Typical x86 PC I/O System



AMD X4 and Intel Nehalem include the north bridge inside the microprocessor, and the south bridge chip of the Intel 975 includes a RAID controller.

Enterprise South Bridge Interface (ESI)

I/O Chips

	Intel 5000P chip set	Intel 975X chip set	AMD 580X CrossFiret
Target segment	Server	Performance PC	Server/Performance PC
Front Side Bus (64 bit)	1066/1333 MHz	800/1066 MHz	—
Memory controller hub ("north bridge")			
Product name	Blackbird 5000P MCH	975X MCH	
Pins	1432	1202	
Memory type, speed	DDR2 FBDIMM 667/533	DDR2 800/667/533	
Memory buses, widths	4 × 72	1 × 72	
Number of DIMMs, DRAM/DIMM	16, 1 GB/2 GB/4 GB	4, 1 GB/2 GB	
Maximum memory capacity	64 GB	8 GB	
Memory error correction available?	Yes	No	
PCIe/External Graphics Interface	1 PCIe x16 or 2 PCIe x	1 PCIe x16 or 2 PCIe x8	
South bridge interface	PCIe x8, ESI	PCIe x8	
I/O controller hub ("south bridge")			
Product name	6321 ESB	ICH7	580X CrossFire
Package size, pins	1284	652	549
PCI-bus: width, speed	Two 64-bit, 133 MHz	32-bit, 33 MHz, 6 masters	—
PCI Express ports	Three PCIe x4		Two PCIe x16, Four PCI x1
Ethernet MAC controller, interface	—	1000/100/10 Mbit	—
USB 2.0 ports, controllers	6	8	10
ATA ports, speed	One 100	Two 100	One 133
Serial ATA ports	6	2	4
AC-97 audio controller, interface	—	Yes	Yes
I/O management	SMbus 2.0, GPIO	SMbus 2.0, GPIO	ASF 2.0, GPIO

AMD X4 includes the north bridge in the microprocessor

Interfacing I/O Devices

- A network protocol defines how a block of data should be communicated on a set of wires.
- This will leaves several other tasks that must be performed to actually cause to be transferred from a device and into the memory address space of some user program.
- This section focuses on these tasks
  - How is a user I/O request transformed into a device command and communicated to the device?
  - How is data actually transferred to or from a memory location?
  - What is the role of the OS?

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

Characteristics of I/O Systems

- The OS guarantees that a user's program accesses only the portions of an I/O devices to which the user has rights.
- The OS provides abstractions for accessing devices by supplying routines that handle low-level devices by a program.
- The OS handles the interrupts generated by I/O devices, just as it handles the exceptions generated by a program.
- The OS tries to provide equitable access to the shared I/O resources, as well as schedules accesses to enhance system throughput.

Three type of Communications

- The OS must be able to give commands to I/O devices.
- The device must be able to notify the OS when the I/O device has completed an operation or has encountered an error.
- Data must be transferred between memory and an I/O device.

I/O Register Mapping

- Memory mapped I/O
  - Registers in the I/O devices are addressed in same space as memory
  - Address decoder distinguishes between them
  - OS uses address translation mechanism to make them only accessible to kernel
- I/O instructions
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86

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

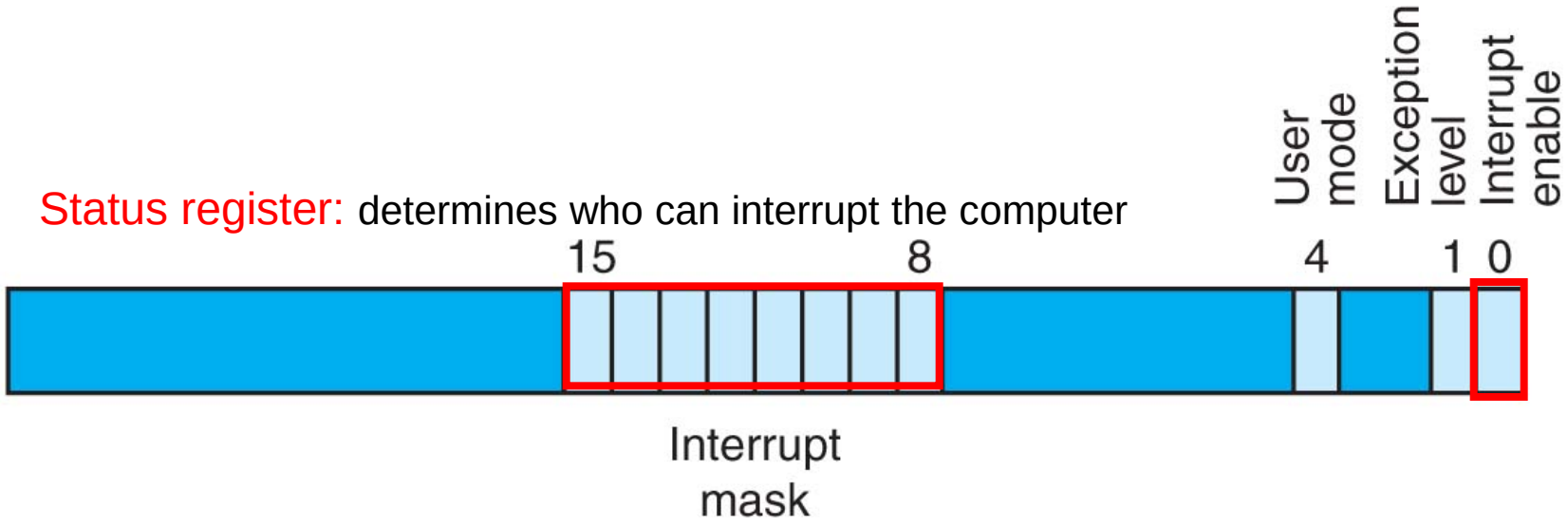
- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception with two important distinctions
  - Asynchronous to instruction execution
    - Can invoke handler between instructions
  - Processor need to get the identity of the device generating the interrupt, as well as its priority.
- A system can use a vectored interrupt or an exception Cause register.
  - When the processor recognizes the interrupt, the device can send either the vector address or a status field to place in the Cause register.

Interrupt Priority Levels

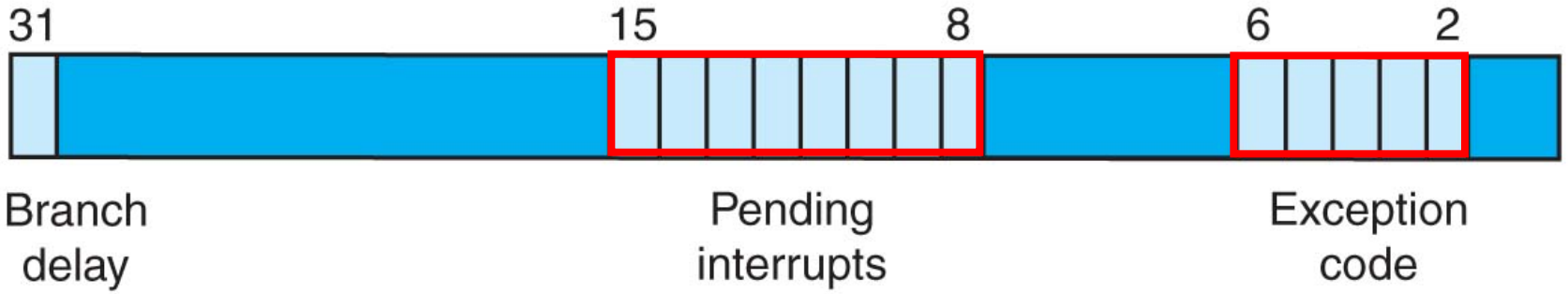
- Most interrupt mechanisms have several levels of priority
  - UNIX OS use 4 to 6 levels
  - I/O interrupts have lower priority than internal exceptions
- MIPS provides the primitives that let the OS implement the policy
  - Key registers are shown in Fig. 6.11
    - Cause and Status registers
  - Interrupt enable=0: no one can interrupt
  - Interrupt mask field in the status register
  - Pending interrupt field in the cause register

Cause and Status Registers

Status register: determines who can interrupt the computer

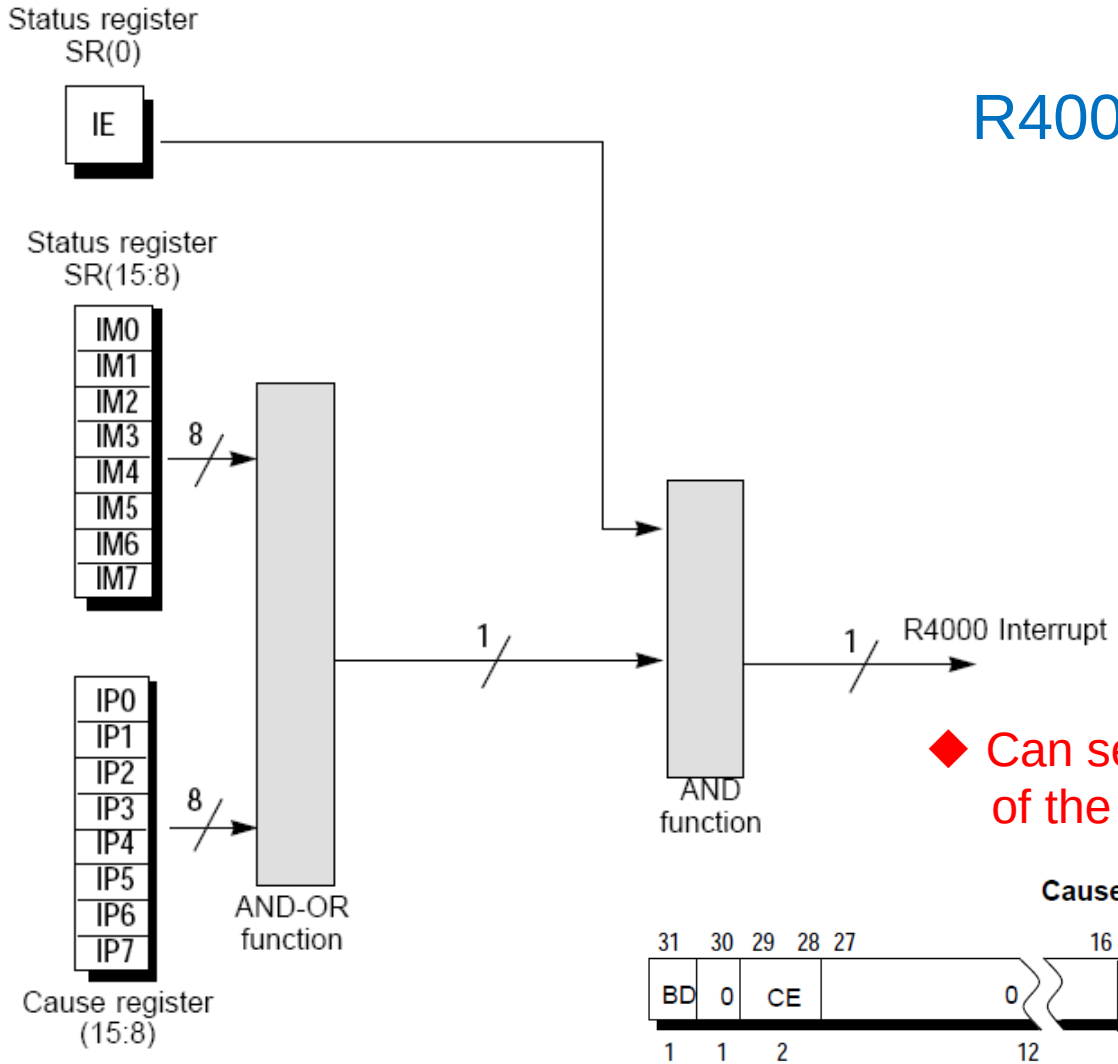


Cause register

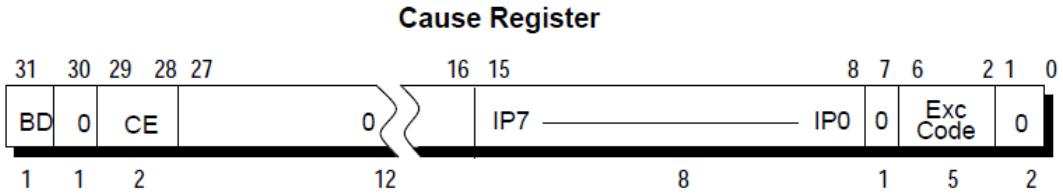


Steps for handling an interrupt

R4000 MIPS processor



- Can set the exception code field of the Cause register



Exception Code

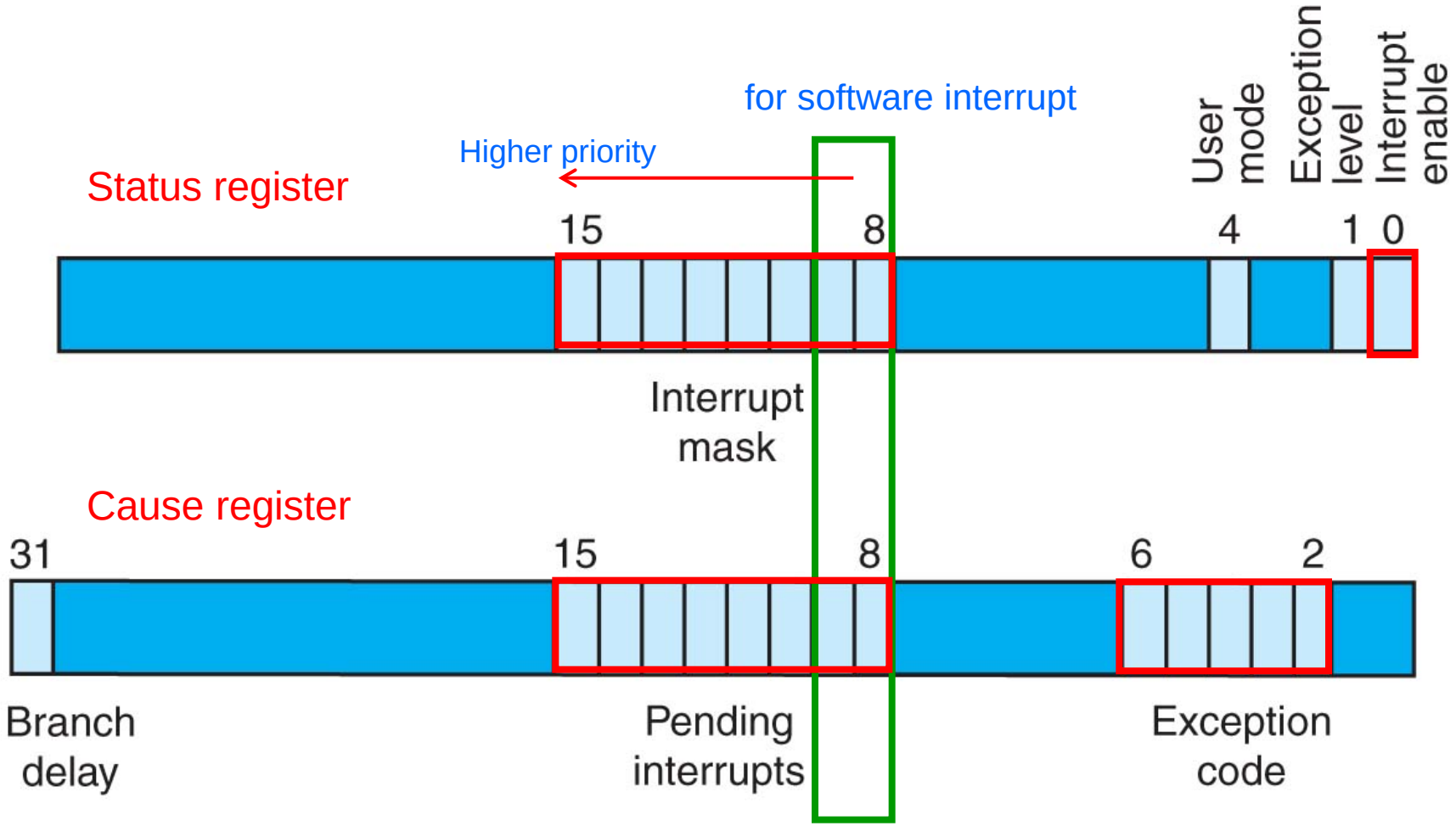
Table 5-6 Cause Register ExcCode Field

Exception Code Value	Mnemonic	Description
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (load or instruction fetch)
3	TLBS	TLB exception (store)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	IBE	Bus error exception (instruction fetch)
7	DBE	Bus error exception (data reference: load or store)
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved instruction exception
11	CpU	Coprocessor Unusable exception
12	Ov	Arithmetic Overflow exception
13	Tr	Trap exception
14	VCEI	Virtual Coherency Exception instruction
15	FPE	Floating-Point exception
16–22	–	Reserved
23	WATCH	Reference to *WatchHi*/*WatchLo* address
24–30	–	Reserved
31	VCED	Virtual Coherency Exception data

Steps for handling an interrupt

1. Logically AND the pending interrupt field (Cause register) and the interrupt mask field (Status register) to see which enabled interrupt could be the culprit. Copies are made of these two registers using mfc0.
2. Select the higher priority of these interrupts. The software convention is that the leftmost is the highest priority.
3. Save the interrupt mask field of the Status register.
4. Change the interrupt mask field to disable all interrupts of equal or lower priority.
5. Save the processor state needed to handle the interrupt
6. To allow higher-priority interrupts, set enable bit of the Status register 1.
7. Call the appropriate interrupt routine
8. Before restoring state, set the interrupt enable bit of the Status register to 0. This also allows you to restore the interrupt mask field.

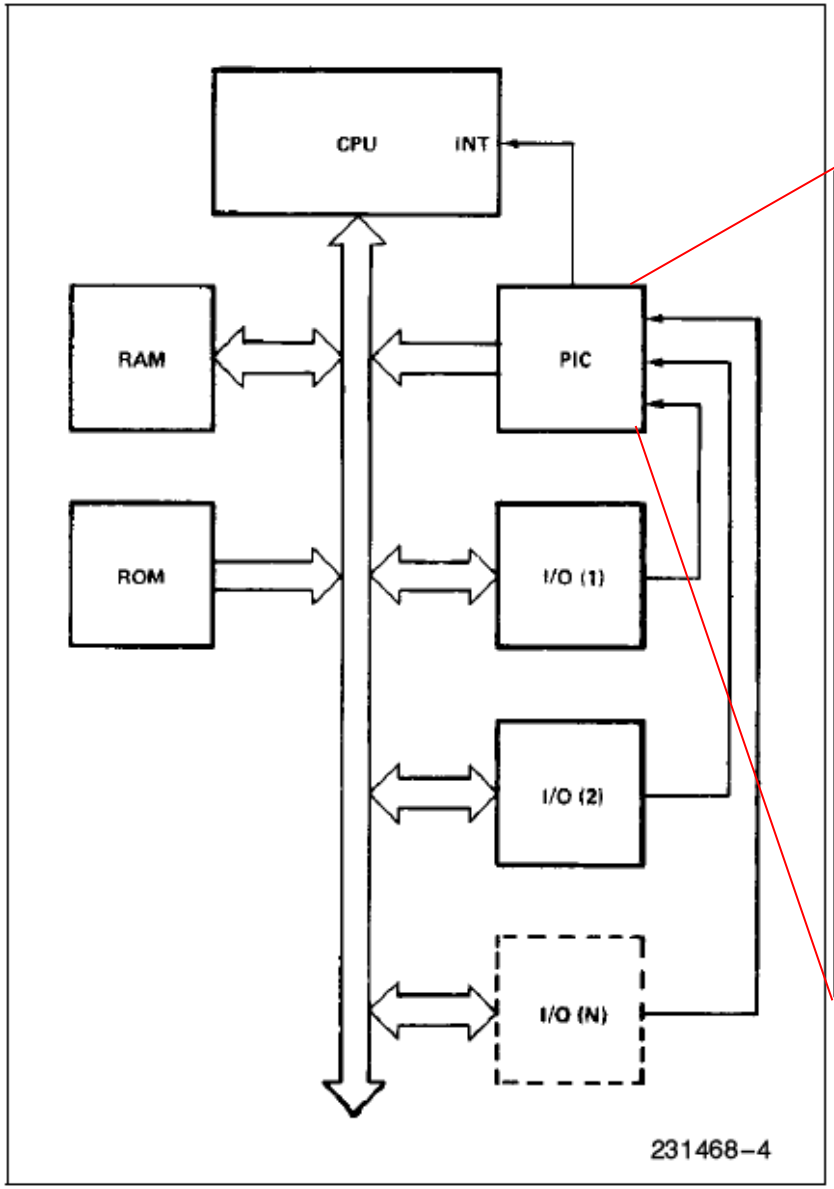
Cause and Status Registers



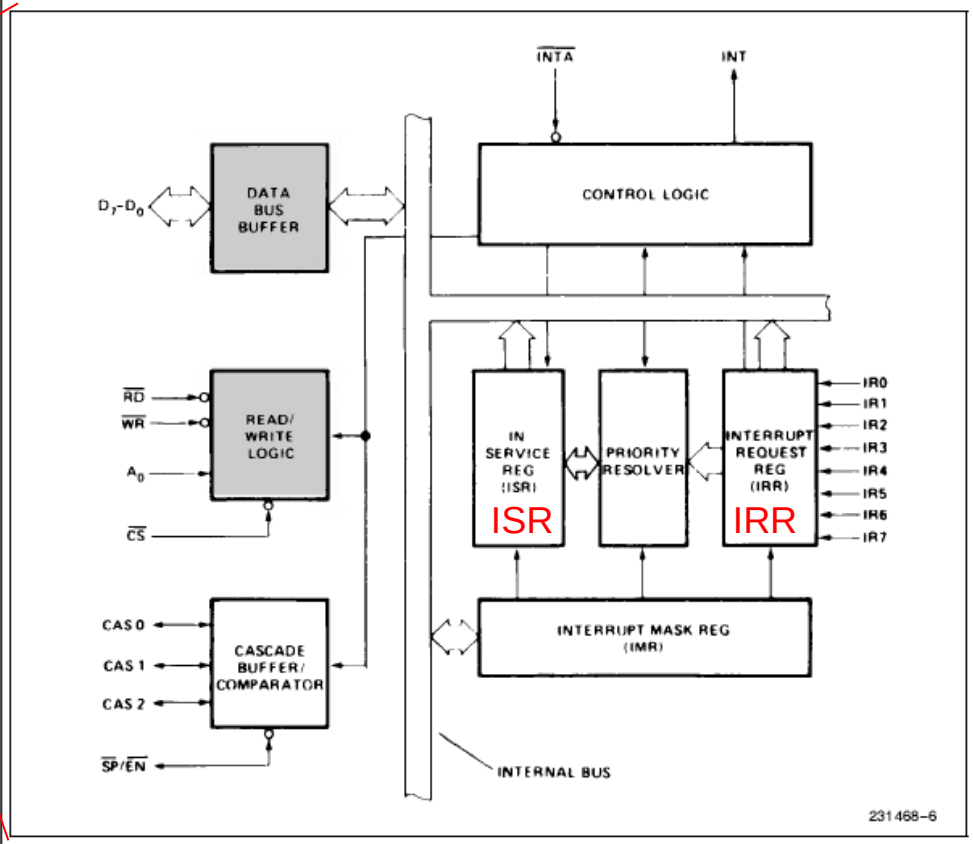
Interrupt Priority Levels (IPLs)

- How do IPLs correspond to these mechanism?
- The IPL is an OS invention.
  - It is stored in the memory of the process, and every process is given an IPL.
- At the lowest IPL, all interrupts are permitted.
- At the highest IPL, all interrupt are blocked.
- Raising and lowering the IPL involves changes to the interrupt mask field of the Status register.

Programmable Interrupt Controller

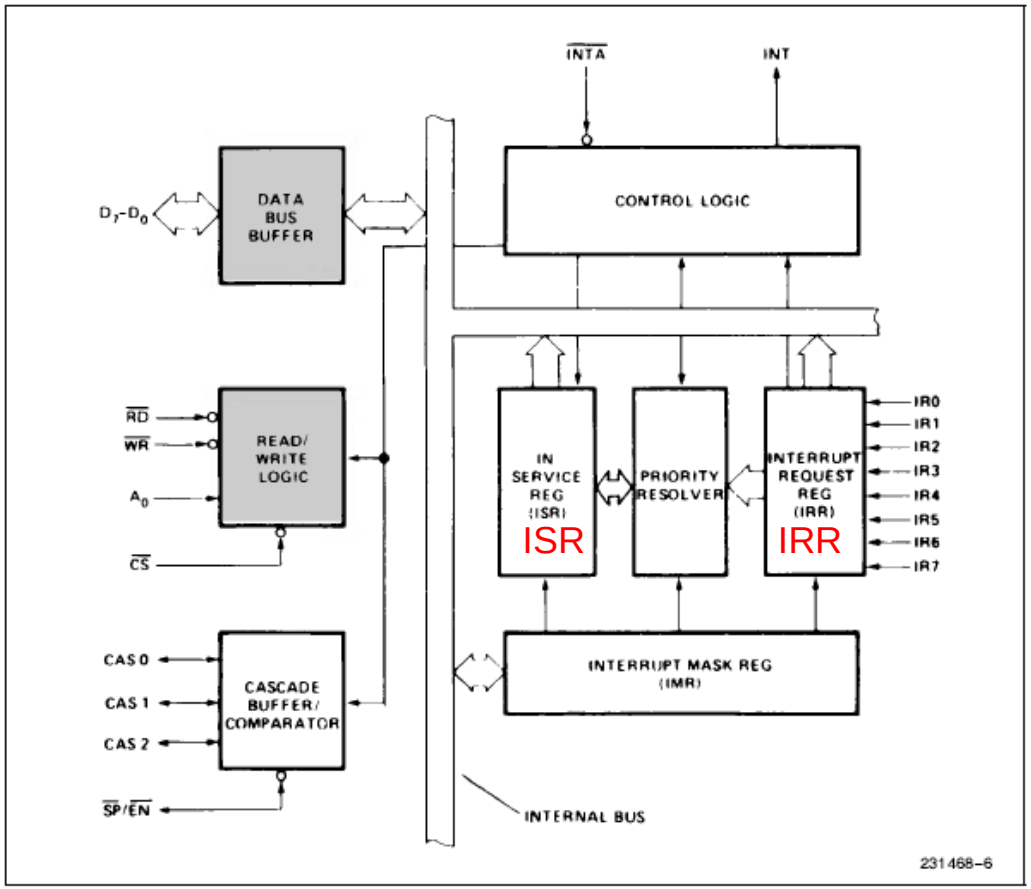


PIC: Intel 8259A



PIC Interrupt Sequence

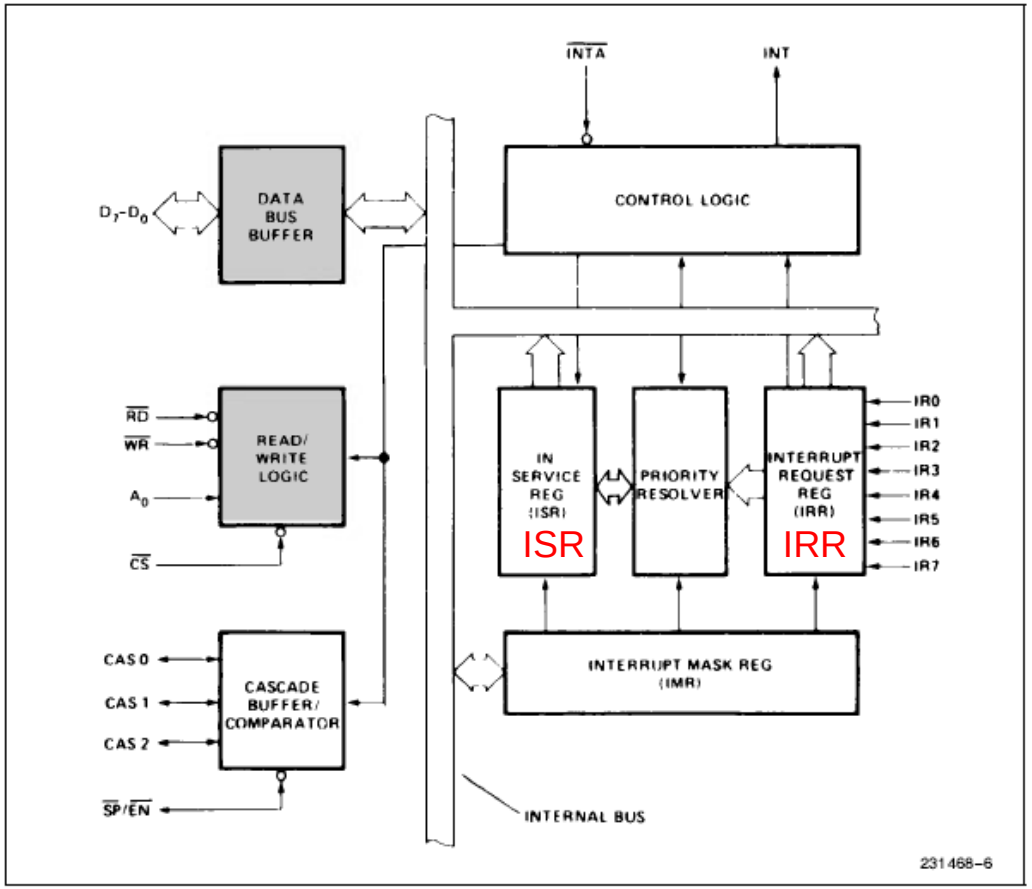
- 1. One or more of the INTERRUPT REQUEST lines are raised high, setting the corresponding IRR bit(s).



- 2. The PIC (8259A) evaluates these requests, and sends an INT to the CPU, if appropriate.
- 3. The CPU acknowledges the INT and responds with an INTA pulse.
- 4. Upon receiving an INTA from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The PIC does not drive the Data Bus during this cycle.

PIC Interrupt Sequence

- 5. The 8086 will initiate a second INTA pulse. During this pulse, the PIC releases an 8-bit pointer (vector address) onto the Data Bus where it is read by the CPU.
- 6. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second INTA pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.



AEOI: automatic end of interrupt

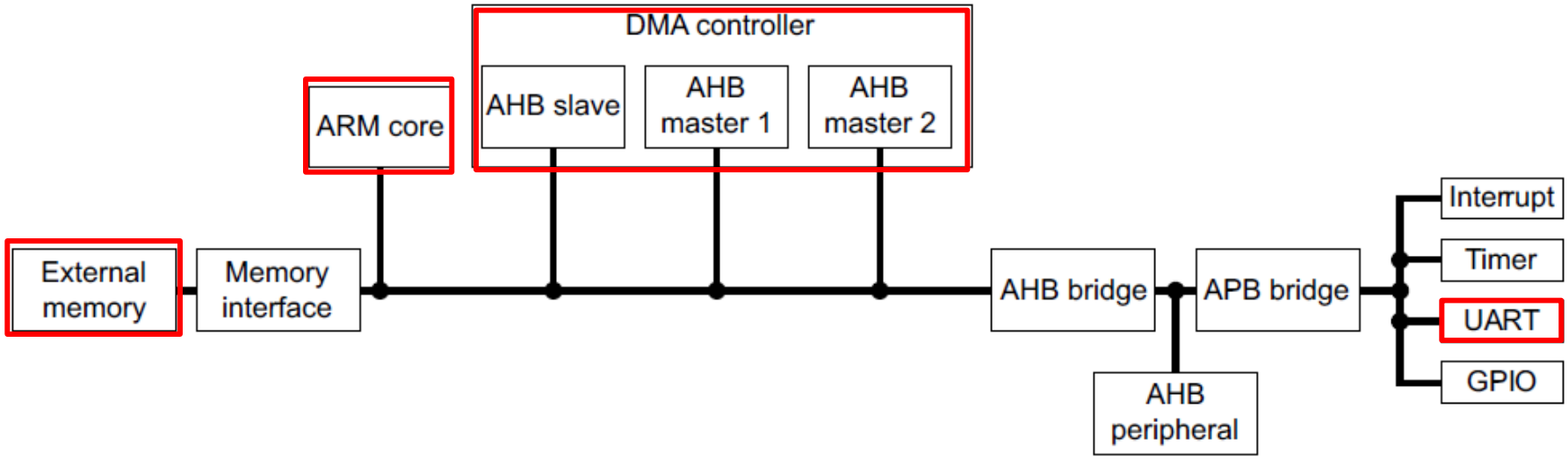
I/O Data Transfer

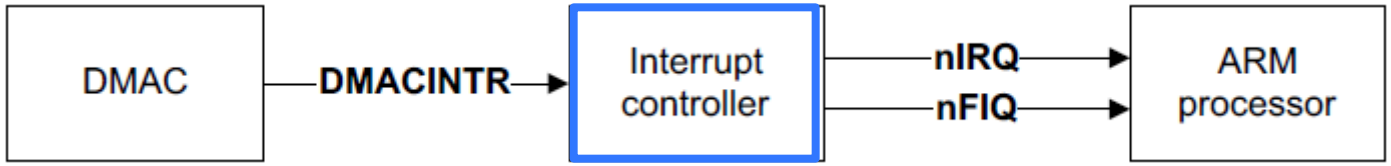
- CPU transfers data between memory and I/O data registers between a device and memory
- Two techniques
  - Polling and interrupt-driven I/O
  - Both work best with lower-bandwidth devices
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - Offloading the processor
  - For high-bandwidth devices
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error

Three steps in a DMA transfer

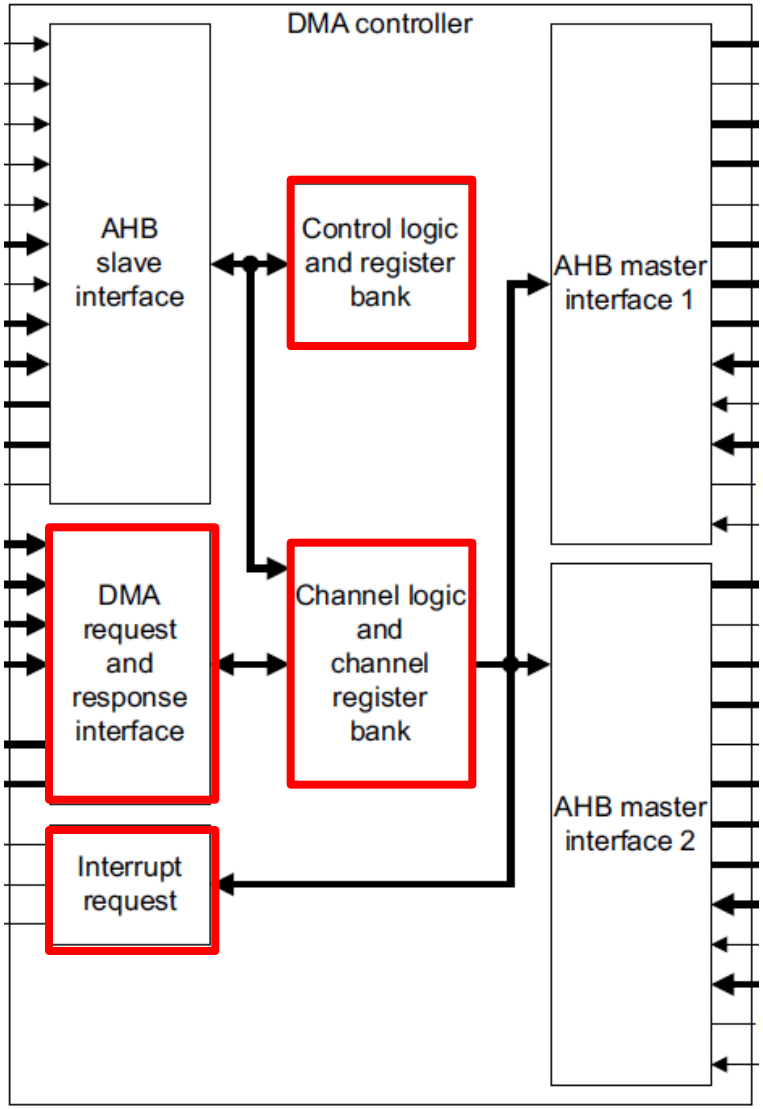
- 1. The processor sets up the DMA by supplying
  - the identity of the device,
  - the operation to perform on the device,
  - the memory address that is the source or destination of the data to be transferred, and
  - the number of bytes to transfer
- 2. The DMA arbitrates for the interconnect and starts the operation on the device.
- 3. Once the DMA transfer is complete, the controller interrupts the processor,
  - which can then determine whether the entire DMA operation completed successfully.

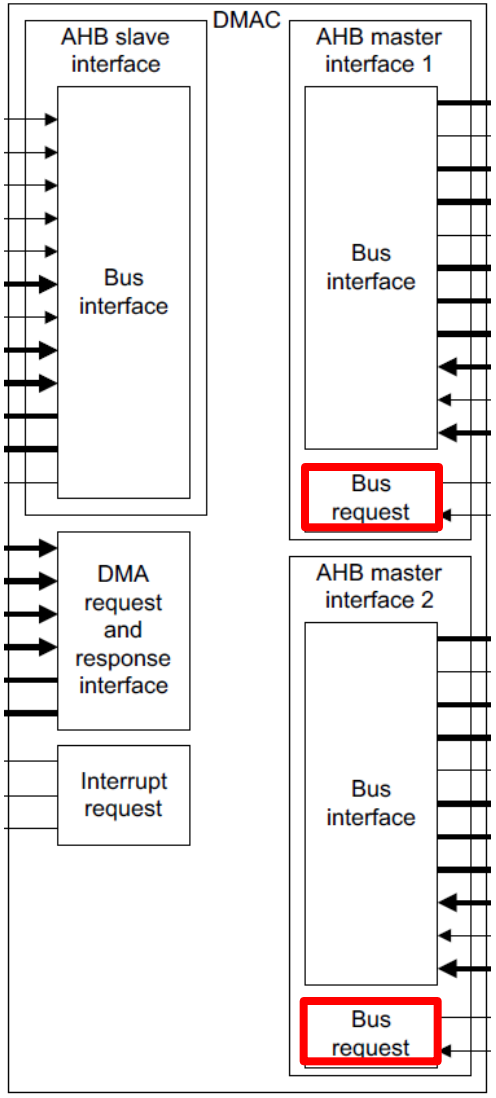
DMA Controller





DMA Controller





Multiple channels in DMAC

- A dedicated channel supports each stream, including source and destination controllers, and a FIFO. This enables better latency than a DMAC with only a single channel shared among several DMA streams.
- DMAC enables the following transactions
  - Memory to memory
  - Memory to a peripheral
  - A peripheral to memory
  - A peripheral to a peripheral
- Each DMA stream provides unidirectional serial transfers for a single source and destination

Bus system & DMA transfers

- There may be multiple DMA devices in a computer system.
  - For example, in a system with a single processor-memory bus and multiple I/O buses, each I/O bus controller will often contain a DMA processor that handles any transfers between a device on the I/O bus and the memory.
- By using caches, the processor can avoid having to access memory most of the time, thereby leaving most of the memory bandwidth free for use by I/O devices.
  - If the processor is contending for memory, it will be delayed when the memory is busy doing a DMA transfer.

I/O Processor

- To further reduce the need to interrupt the processor and occupy it in handling an I/O request that may involve doing several actual operations, the I/O controller can be made more intelligent.
- Specialized processors that basically execute a series of I/O operations, called an I/O program.
  - Can be general purpose microprocessors

DMA and Memory System

- Without DMA, all accesses to memory go through address translation and cache access
- With DMA, there is another path to the memory system – one that does no go through the address translation
- This problem are usually solved with a combination of hardware techniques and software support.
- Difficulties in having DMA in a virtual memory system
  - Pages have both a physical and a virtual address.

DMA/VM Interaction

- OS uses virtual addresses for memory
  - DMA blocks may not be contiguous in physical memory → DMA becomes not efficient
- Should DMA use virtual addresses?
  - Would require controller to do translation→ complex
- If DMA uses physical addresses
  - May need to break transfers into page-sized chunks
  - Or chain multiple transfers (chained DMA transactions)
  - Or allocate contiguous physical pages for DMA
- Whichever method is used, the OS must still cooperate by not remapping pages while a DMA transfer involving the page is in progress.

DMA/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes stale
- If write-back cache has dirty block, and DMA reads memory block → Reads stale data
- Need to ensure cache coherence
  - Route the I/O activity through the cache
    - Degrade performance
  - OS selectively invalidates the cache block for an I/O read or force write-backs to occur for an I/O write (often called cache flushing)
  - HW mechanism for selectively flushing cache entries
  - Or use non-cacheable memory locations for I/O

Measuring I/O Performance

- How should we compare I/O systems?
- I/O performance depends on
  - Hardware: CPU, memory, controllers, buses
  - Software: operating system, database management system, application
  - Workload: request rates and patterns
- I/O system design can trade-off between response time and throughput
  - Measurements of throughput often done with constrained response-time

A confusion point

- The transfer rate depends on the clock rate
- 1 GHz= 10^9 cycles per second
- 1 GB = 1,000,000,000 bytes in I/O systems

- 1 GB = 2^{30} bytes = 1,073,741,824 bytes in main memory

- The difference need to convert between base 10 (1K=1000) and base 2 (1k=1024)

Transaction Processing

- Transaction processing (TP): A typical application that involves handling small short operations (called transactions) that typically require both I/O and computation.
  - Small data accesses to a DBMS
  - TP processing applications typically have both response time requirements and a performance measurement based on the throughput of transactions
  - Interested in I/O rate, not data rate
- I/O rate: performance measure of I/Os per unit time, such as reads per second
- Data rate: performance measure of bytes per unit times, such as GB/second

Transaction Processing Benchmarks

- Measure throughput
  - Subject to response time limits and failure handling
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Overall cost per transaction
- Transaction Processing Council (TPC) benchmarks (www.tcp.org)
  - TPC-APP: B2B application server and web services
  - TCP-C: on-line order entry environment
  - TCP-E: on-line transaction processing for brokerage firm
  - TPC-H: decision support — business oriented ad-hoc queries
- All the TPC benchmarks measure performance in transactions per second
  - Measured only when a response time limit is met

File System & Web I/O Benchmarks

- SPEC System File System (SFS)
  - Synthetic workload for NFS server, based on monitoring real systems
  - Results
    - Throughput (operations/sec)
    - Response time (average ms/operation)
- SPEC Web Server benchmark
  - Measures simultaneous user sessions, subject to required throughput/session
  - Three workloads: Banking, Ecommerce, and Support

Designing I/O Systems

- Two primary types of specifications
  - Latency constraints
  - Bandwidth constraints
- Knowledge of the traffic patterns affects the design and analysis
  - For time-critical operations
- Determining the latency on an unloaded system is relatively easy simple
- Finding the average latency under a load is much harder. Tackled either
  - by queuing theory or
  - by simulation

Designing I/O Systems

- Another typical problem designers face
  - Meeting a set of bandwidth constraints given a workload
- Given a partially configure I/O system, balancing the system to maintain the maximum bandwidth achievable is a simplified problem of the first.
- The general approaches to designing an I/O system
  - Find the weakest link in the I/O system
  - Configure the component to sustain the required bandwidth
  - Determine the requirements for the rest of the system and configure them to support the bandwidth

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: a misnomer
  - No redundancy (“AID”?)
    - Just stripe data over multiple disks
  - But it does improve performance
  - Striping: allocation of logically sequential blocks to separate disks

RAID 1 & 2

- RAID 1: Mirroring (shadowing)
  - Mirroring: writing the identical data to multiple disks to increase data availability
  - N + N disks, replicate data: expensive
    - Write data to both data disk and mirror disk
    - On disk failure, read from mirror
- RAID 2: Error correcting code (ECC)
  - N + E disks (e.g., 10 + 4)
    - Split data at bit level across N disks
    - Generate E-bit ECC
  - Too complex, not used in practice

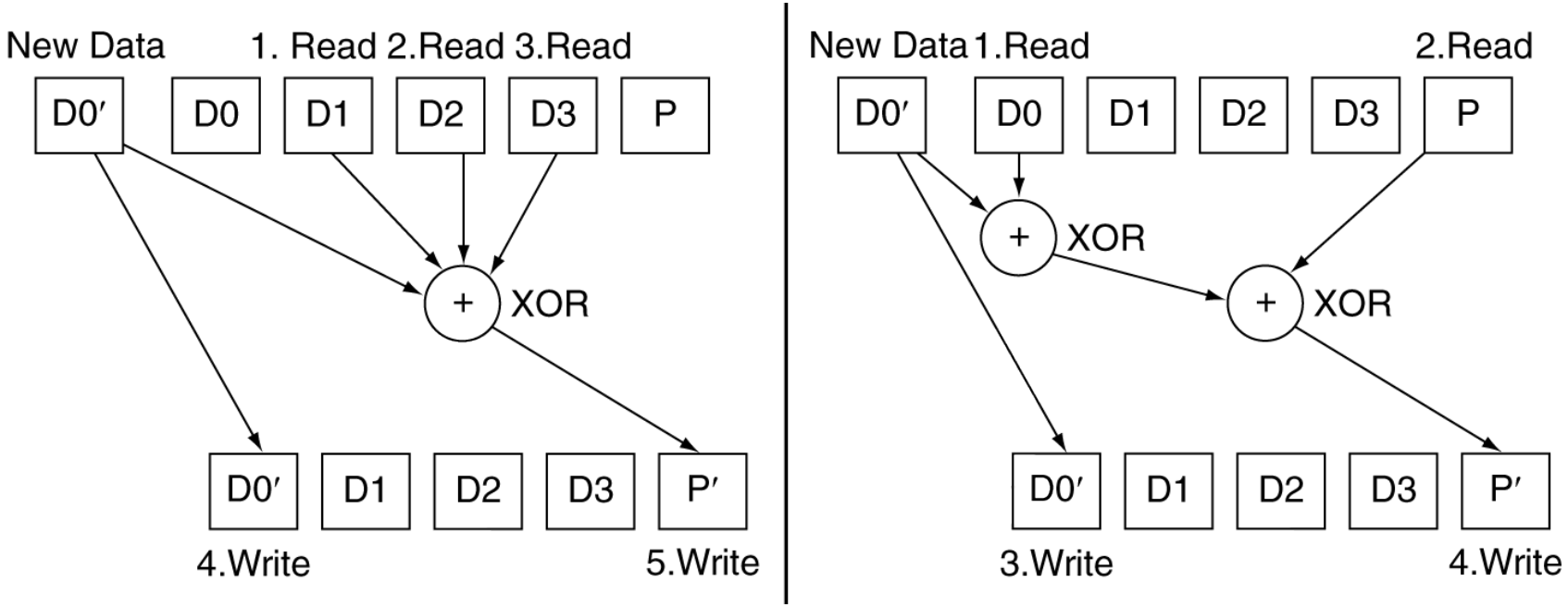
RAID 3: Bit-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at byte level
  - A redundant disk stores parity
  - Read access
    - Read all disks
  - Write access
    - Generate new parity and update all disks
  - On failure
    - Use parity to reconstruct missing data
- Not widely used

RAID 4: Block-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at block level
  - A 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 missing data
- Not widely used

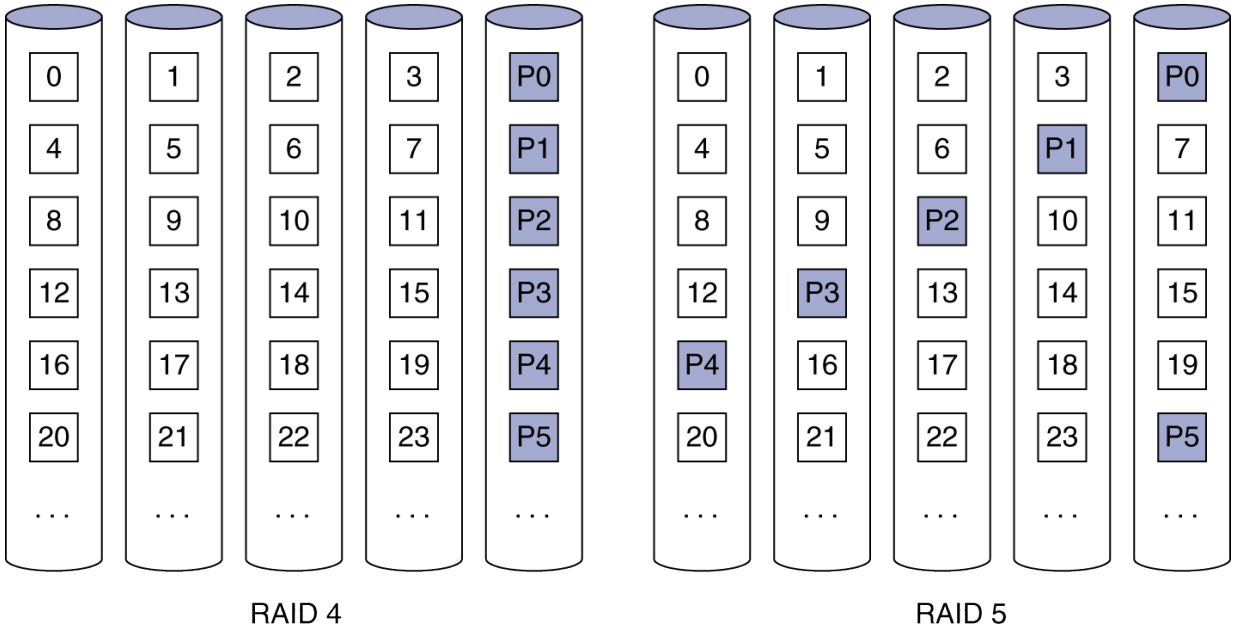
RAID 3 vs RAID 4



The RAID 4 shortcut on the right reads the old value D0 and compares it to the new value D0' to see which bits will change. You then read the old parity P and then change the corresponding bits to form P'. The logical function exclusive OR does exactly what we want. This example replaces three disk reads (D1, D2, D3) and two disk writes (D0', P') involving all the disks for two disk reads (D0, P) and two disk writes (D0', P'), which involve just two disks. Increasing the size of the parity group increases the savings of the shortcut. RAID 5 uses the same shortcut.

RAID 5: Distributed Parity

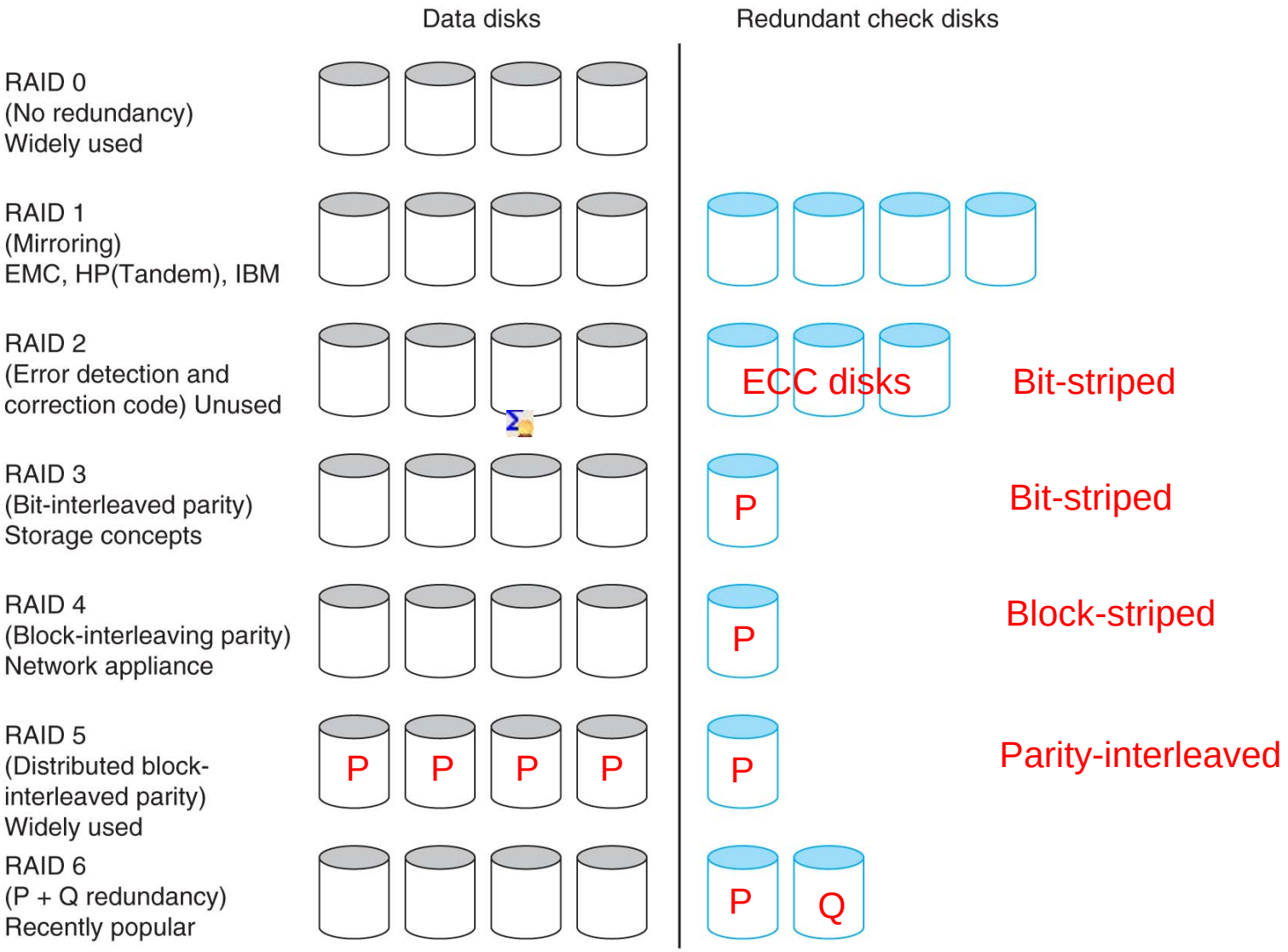
- N + 1 disks
  - Like RAID 4, but parity blocks distributed across disks
    - Avoids parity disk being a bottleneck
- Widely used



RAID 6: P + Q Redundancy

- N + 2 disks
  - Like RAID 5, but two lots of parity (P & Q)
  - Greater fault tolerance through more redundancy

RAID


Data disks
Redundant check disks
RAID 0
(No redundancy)
Widely used
RAID 1
(Mirroring)
EMC, HP(Tandem), IBM
RAID 2
(Error detection and
correction code) Unused
ECC disks
Bit-striped
RAID 3
(Bit-interleaved parity)
Storage concepts
P
Bit-striped
RAID 4
(Block-interleaving parity)
Network appliance
P
Block-striped
RAID 5
(Distributed block-
interleaved parity)
Widely used
P P P P
P
Parity-interleaved
RAID 6
(P + Q redundancy)
Recently popular
P
Q

RAID Summary

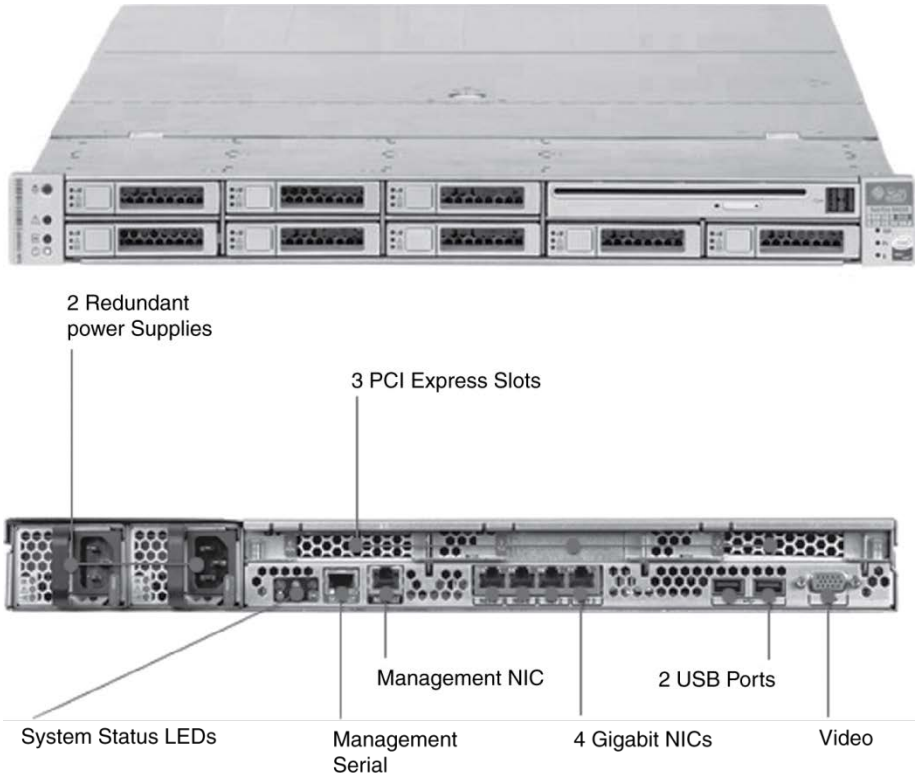
- RAID can improve performance and availability
  - High availability requires hot swapping
- Assumes independent disk failures
  - Too bad if the building burns down!
- See “Hard Disk Performance, Quality and Reliability”
  - http://www.pcguide.com/ref/hdd/perf/index.htm

Server Computers

- Applications are increasingly run on servers
  - Web search, office apps, virtual worlds, …
- Requires large data center servers
  - Multiple processors, networks connections, massive storage
  - Space and power constraints
- Server equipment built for 19” racks
  - 19” wide (482.6 mm), depth varies
  - Multiples of 1.75” (1U) high

19-inch rack with 42 1U servers

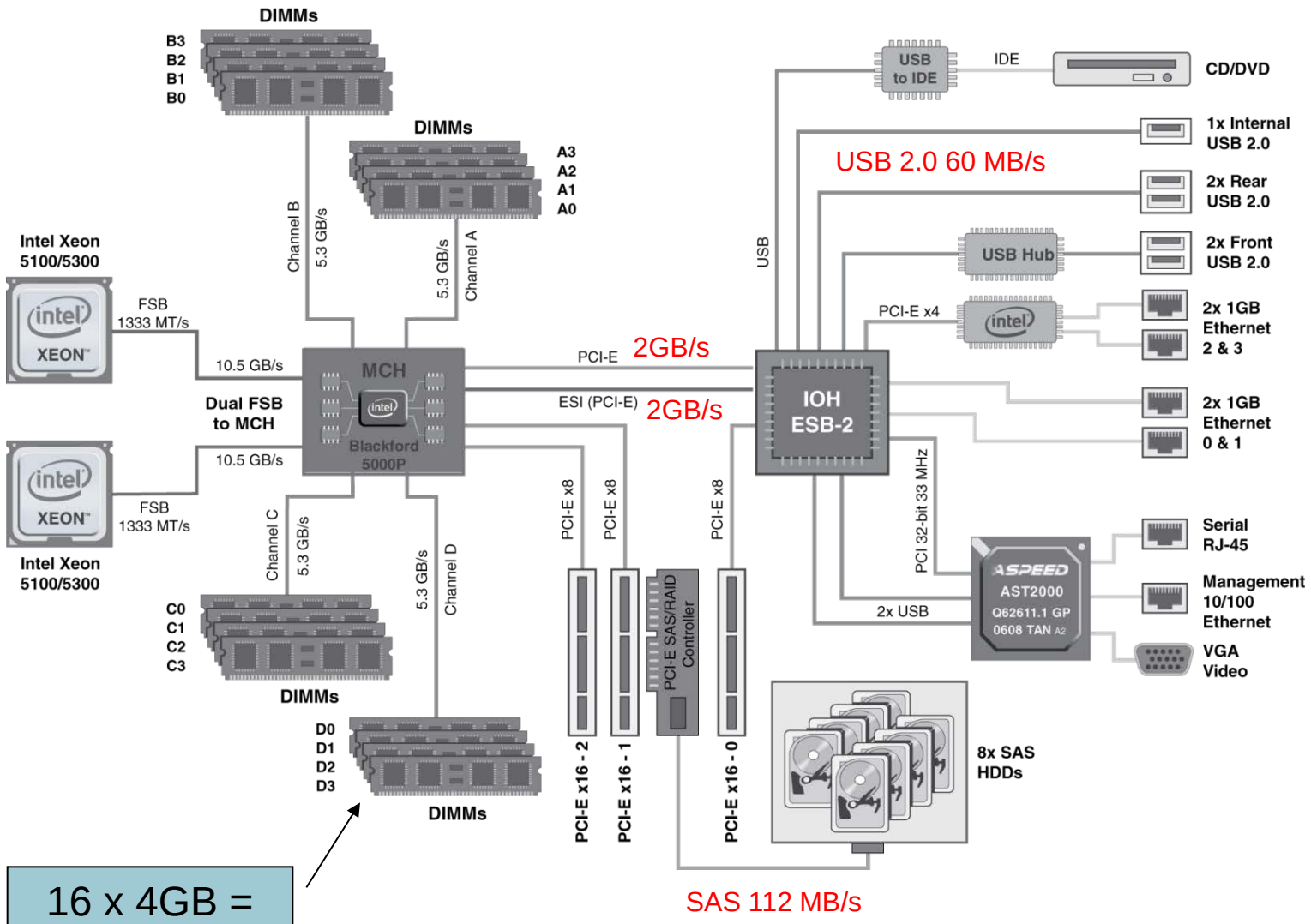
Sun Fire x4150 1U server



Sun Fire x4150

- The 1U box contains
  - Eight 2.66 GHz processors, spread across two sockets (2 Intel Xeon 5345)
  - 64 GB of DDR2-667 DRAM, spread across 16 4GB fully buffered DIMMs (FBDIMMs)
  - Eight 15,000 RPM 73 GB SAS 2.5-inch disk drives
  - 1 RAID controller (supporting RAID 0, RAID 1, RAID5, RAID 6)
  - Four 10/100/1000 Ethernet ports
  - Three PCI Express x8 ports
  - 4 external and 1 internal USB ports

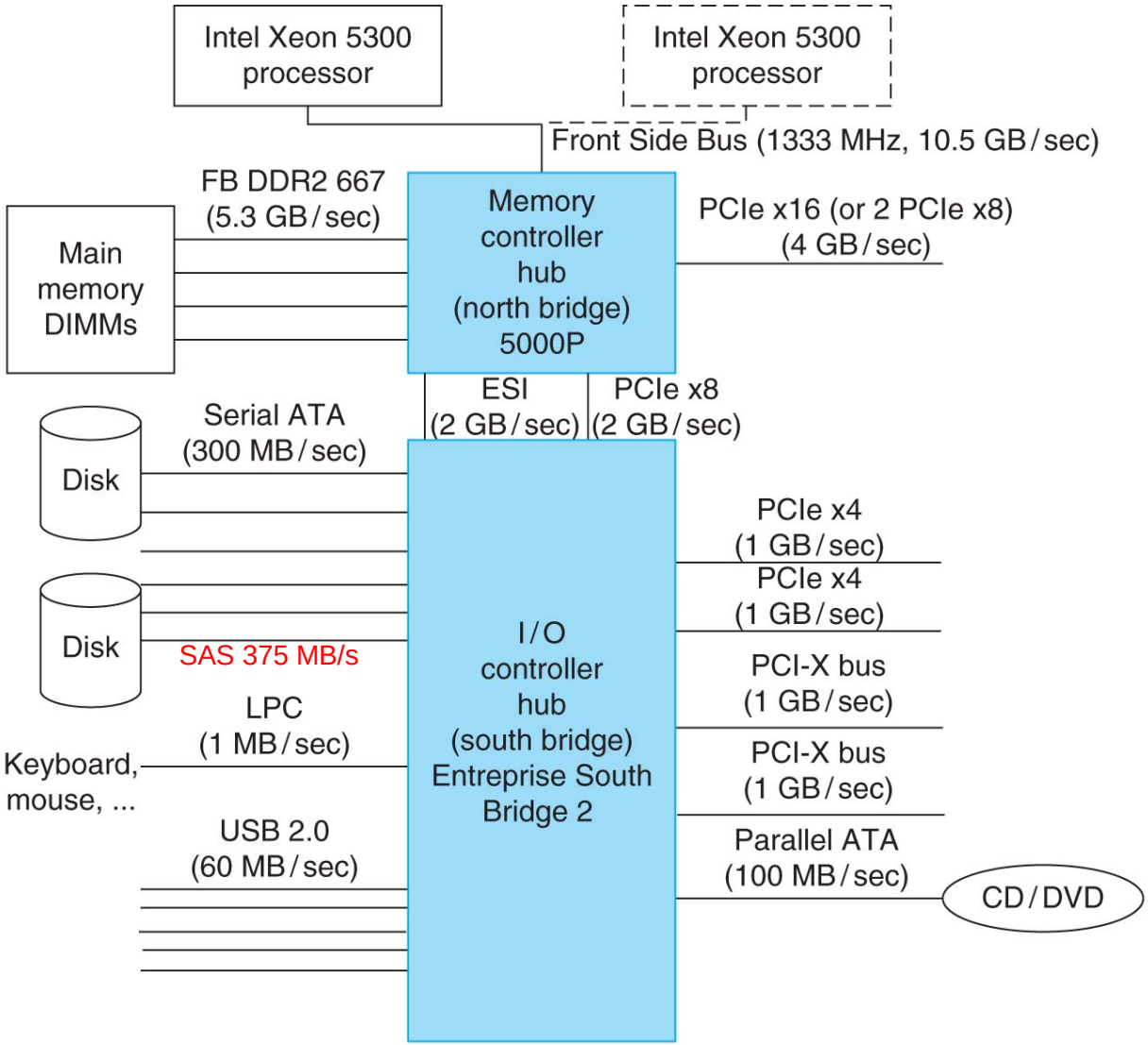
Sun Fire x4150 1U server



4 cores each

16 x 4GB = 64GB DRAM

I/O Systems on an Intel Server



Intel 5000P Chip Set

	Intel 5000P chip set	Intel 975X chip set	AMD 580X CrossFiret
Target segment	Server	Performance PC	Server/Performance PC
Front Side Bus (64 bit)	1066/1333 MHz	800/1066 MHz	—
	Memory controller hu (“north bridge”)		
Product name	Blackbird 5000P MCH	975X MCH	
Pins	1432	1202	
Memory type, speed	DDR2 FBDIMM 667/533	DDR2 800/667/533	
Memory buses, widths	4 × 72	1 × 72	
Number of DIMMs, DRAM/DIMM	16, 1 GB/2 GB/4 GB	4, 1 GB/2 GB	
Maximum memory capacity	64 GB	8 GB	
Memory error correction available?	Yes	No	
PCIe/External Graphics Interface	1 PCIe x16 or 2 PCIe x	1 PCIe x16 or 2 PCIe x8	
South bridge interface	PCIe x8, ESI	PCIe x8	
	I/O controller hub (“south bridge”)		
Product name	6321 ESB	ICH7	580X CrossFire
Package size, pins	1284	652	549
PCI-bus: width, speed	Two 64-bit, 133 MHz	32-bit, 33 MHz, 6 masters	—
PCI Express ports	Three PCIe x4		Two PCIe x16, Four PCI x1
Ethernet MAC controller, interface	—	1000/100/10 Mbit	—
USB 2.0 ports, controllers	6	8	10
ATA ports, speed	One 100	Two 100	One 133
Serial ATA ports	6	2	4
AC-97 audio controller, interface	—	Yes	Yes
I/O management	SMbus 2.0, GPIO	SMbus 2.0, GPIO	ASF 2.0, GPIO

240-pin fully buffered DIMM

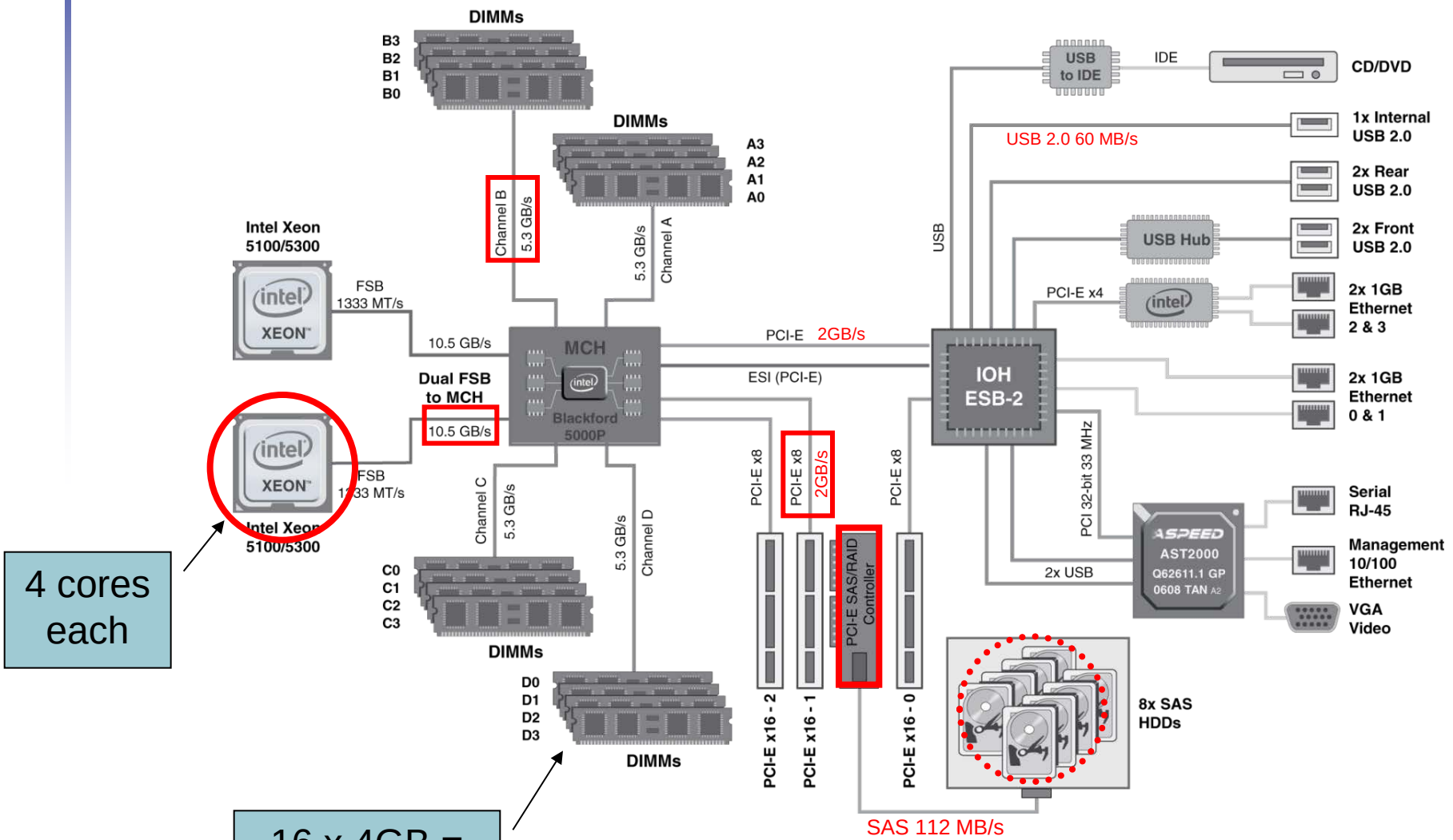
SAS Disk in Sun Fire x4150

Characteristics	Seagate ST33000655SS	Seagate ST31000340NS	Seagate ST973451SS	Seagate ST9160821AS
Disk diameter (inches)	3.50	3.50	2.50	2.50
Formatted data capacity (GB)	147	1000	73	160
Number of disk surfaces (heads)	2	4	2	2
Rotation speed (RPM)	15,000	7200	15,000	5400
Internal disk cache size (MB)	16	32	16	8
External interface, bandwidth (MB/sec)	SAS, 375	SATA, 375	SAS, 375	SATA, 150
Sustained transfer rate (MB/sec)	73–125	105	79–112	44
Minimum seek (read/write) (ms)	0.2/0.4	0.8/1.0	0.2/0.4	1.5/2.0
Average seek read/write (ms)	3.5/4.0	8.5/9.5	2.9/3.3	12.5/13.0
Mean time to failure (MTTF) (hours)	1,400,000 @ 25°C	1,200,000 @ 25°C	1,600,000 @ 25°C	—
Annual failure rate (AFR) (percent)	0.62%	0.73%	0.55%	—
Contact start-stop cycles	—	50,000	—	>600,000
Warranty (years)	5	5	5	5
Nonrecoverable read errors per bits read	<1 sector per 10^{16}	<1 sector per 10^{15}	<1 sector per 10^{16}	<1 sector per 10^{14}
Temperature, shock (operating)	5°–55°C, 60 G	5°–55°C, 63 G	5°–55°C, 60 G	0°–60°C, 350 G
Size: dimensions (in.), weight (pounds)	1.0" × 4.0" × 5.8", 1.5 lbs	1.0" × 4.0" × 5.8", 1.4 lbs	0.6" × 2.8" × 3.9", 0.5 lbs	0.4" × 2.8" × 3.9", 0.2 lbs
Power: operating/idle/ standby (watts)	15/11/—	11/8/1	8/5.8/—	1.9/0.6/0.2
GB/cu. in., GB/watt	6 GB/cu.in., 10 GB/W	43 GB/cu.in., 91 GB/W	11 GB/cu.in., 9 GB/W	37 GB/cu.in., 84 GB/W
Price in 2008, $/GB	~ $250, ~ $1.70/GB	~ $275, ~ $0.30/GB	~ $350, ~ $5.00/GB	~ $100, ~ $0.60/GB

I/O System Design Example

- A Sun Fire x4150 system with
  - Workload: 64KB disk reads
    - Each I/O op uses 200,000 user-code instructions and
    - OS averages 100,000 OS instructions per I/O operation
  - Each CPU: 10^9 instructions/sec
- Find the maximum sustainable I/O rate for a fully loaded Sun Fire x4150 for random reads and sequential reads
  - Assume that the reads always be done on an idle disk and that the RAID controller is not be bottleneck.
  - FSB: 10.6 GB/sec peak
  - DRAM DDR2 667MHz: 5.336 GB/sec (x4)
  - PCI-E 8× bus: 8 × 250MB/sec = 2GB/sec
  - Disks: 15,000 rpm, 2.9ms avg. seek time, 112MB/sec sustained transfer rate
- What I/O rate can be sustained?
  - For random reads, and for sequential reads

Sun Fire x4150 1U server

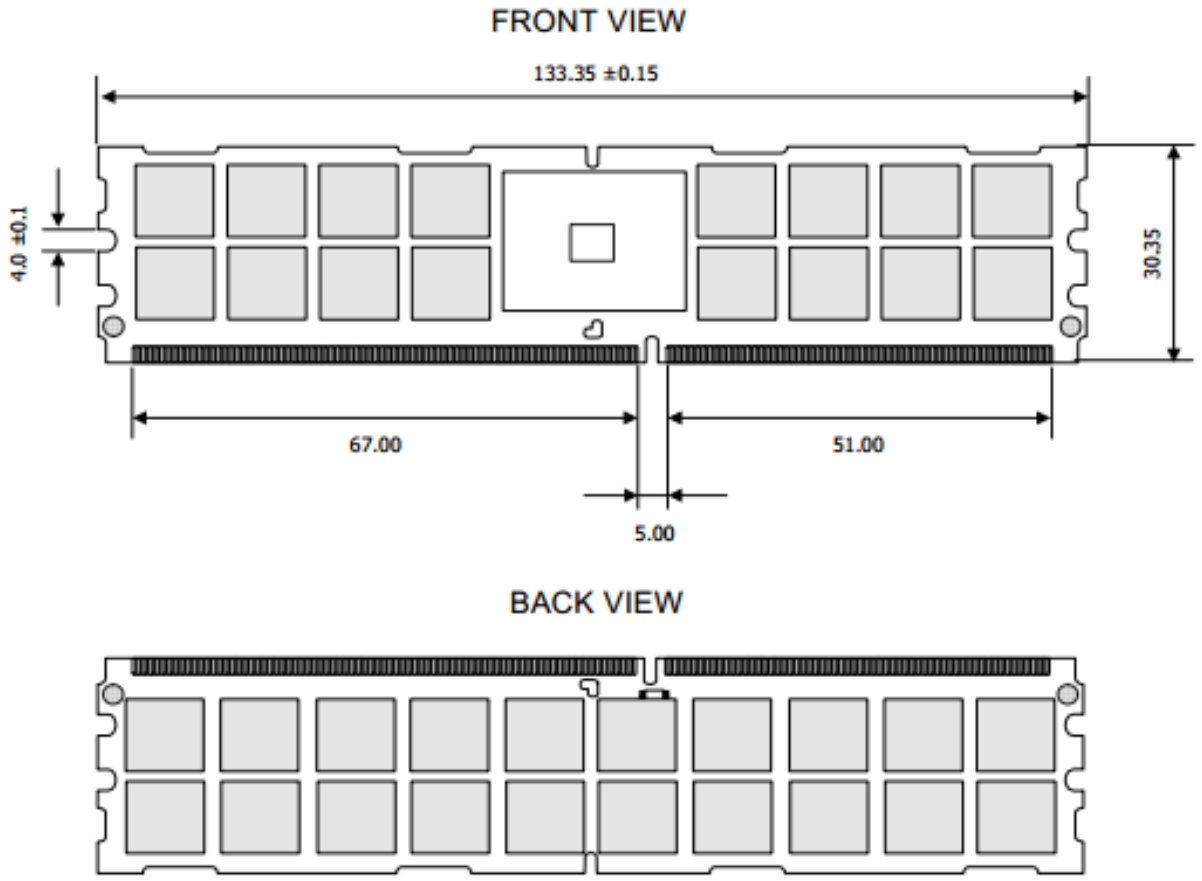


FBDIMM/DDR2 SDRAM

- 240-pin DDR2 fully buffered dual in-line memory module with ECC to detect and report channel errors to host memory controller
- 8 banks
- 4 GB = 256 M x 72 bit (including ECC)

  = 256 M x 4 x 2 x 18 packages
- A burst of eight = 72 x 8 = 64B data + 8B ECC

FBDIMM/DDR2 SDRAM



Design Example (cont)

- I/O rate for CPUs
  - Per core: 10^9/(100,000 + 200,000) = 3,333 IOPS
  - 8 cores: 26,667 IOPS
- Random reads, I/O rate for disks
  - Assume actual seek time is average/4
  - Time/op = seek + rotational latency + transfer
    = 2.9ms/4 + 4ms/2 + 64KB/(112MB/s) = 3.3ms
  - 303 IOPS per disk, 2424 IOPS for 8 disks
- Sequential reads
  - 112MB/s / 64KB = 1750 IOPS per disk
  - 14,000 IOPS for 8 disks

Design Example (cont)

- PCI-E I/O rate
  - 2GB/sec / 64KB = 31,250 ops/sec
- DRAM I/O rate per DIMM
  - 16 DIMM in a fully configured x4150
  - 5.336 GB/sec / 64KB = 83,375 ops/sec
- FSB I/O rate
  - Assume we can sustain half the peak rate
  - 5.3 GB/sec / 64KB = 81,540 IOPS per FSB
  - 163,080 IOPS for 2 FSBs
- Weakest link: disks
  - 2424 random reads per second
  - 14,000 sequential reads per second
  - Other components have ample headroom to accommodate these rates

Fallacy: Disk Dependability

- The rated mean time to failure of disks is as 1.2 M hours (140yr), so disks practically never fail
  - because the lifetime of a disk is 5 years
- Wrong!: this is the mean time to failure
  - What is the distribution of failures?
    - Exponential distribution for constant failure rare
    - Weibull distribution : cdf $R(x;k,\lambda) = 1 - EXP(-(x/\lambda)^k)$
      - $0 < k < 1$: the failure rate decreases over time (infant mortality)
      - $k = 1$: the failure rate is constant
      - $1 < k$: the failure rate increases over time (wear or aging)
  - What if you have 1000 disks
    - How many will fail per year? 7.3 disks

$$\text{Annual Failure Rate (AFR)} = \frac{1000 \text{ disks} \times 8760 \text{ hrs/disk}}{1200000 \text{ hrs/failure}} = 0.73\%$$

1 year= 8760 hrs

Fallacies

- Disk failure rates are as specified
  - No!
  - Studies of failure rates in the field for 100,000 ATA and SCSI disks
    - AFR = 2% to 4% (measured) vs. 0.6% to 0.8% (spec)
  - Another study for more than 100,000 ATA disks
    - AFR= 1.7% (first year) to 8.6% (third year) vs. 1.5% (quoted)
  - Why?

Fallacies

- A 1GB/s interconnect transfers 1GB in one sec
  - You cannot use 100% of any bus bandwidth
    - 70% ~ 80% of the peak bandwidth: fortunate
  - Why you cannot use 100% of a bus
    - Time to send address or acknowledge signals
    - Stalls while waiting to use a busy
  - But what's a GB?
  - For bandwidth, use 1GB = 10^9 B
  - For storage, use 1GB = 2^{30} B = 1.075×10^9 B
  - So 1GB/sec is 0.93GB in one second
    - About 7% error

Pitfall

- Trying to provide features only within the network versus end to end
  - The concern is providing at low level features that can only be accomplished at the highest level, thus only partially satisfying the communication demand.
- End-to-end argument
  - The function in question can completely and correctly be specified only with the knowledge and help of the application standing at the endpoints of the communication system.
  - Therefore, providing that questioned function as a feature of the communication system itself is not possible.
- Example: a MIT network that used several gateways, each of which added a checksum from one gateway to the next.
  - The application programmers assumed the checksum guaranteed accuracy, incorrectly believing that the message was protected while stored in the memory of each gateway.
  - One gateway developed a transient failure that swapped one pair of bytes per million bytes transferred. – wrong!

Pitfall: intermediate checks vs end-to-end

- Over time the source code of one OS was repeatedly passed through the gateway, thereby corrupting the code.
  - The only solution was to correct the infected source files by comparing to paper listings and repairing the code by hand.
- Had the checksums been calculated and checked by the application running on the end system, safety would have been assured.
- There is a useful role for intermediate checks, if end-to-end checking is available.
  - End-to-end checking: can show that something is broken between two ends
  - Intermediate checks: can discover which component is broken
  - You need both for repair

Pitfall: Offloading to I/O Processors

- Offloading expecting to improve performance without a careful analysis
- Overhead of managing an I/O request may dominate when intelligent interfaces are used
  - Would be quicker to do small operation on the CPU
- Performance falls when the I/O processor has much lower performance than the main processor.
  - Since the I/O processor is supposed to be simpler
- Consequently, a small amount of main processor time is replaced with a large amount of I/O processor time.

Wheel of Reincarnation ?

- Making an I/O processor faster makes it into a major system component
  - An I/O processor might need its own coprocessors!
- Myer and Sutherland: they eventually noticed that they were caught in a loop of continuously increasing the power of an I/O processor until it needed its own simpler coprocessor.
  - Display processor added graphic features came to resemble a full-fledged processor

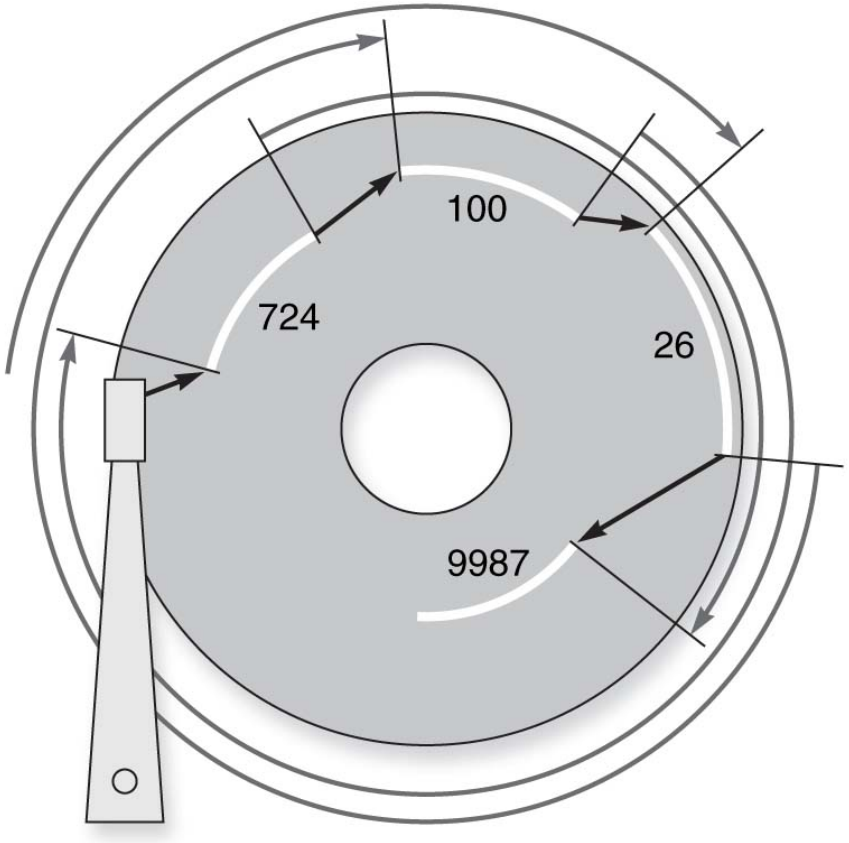
Pitfall: Using Tape to back up disks

- This is both a fallacy and a pitfall
- History: Which are useful backs (magnetic tapes vs. disks)
  - longer access time
  - can be used per reader (removable spool)
  - higher density
  - 10x to 100x cheaper than disks
- Tape readers must read the last 4 generations of tapes
- Desktop owners generally do not backup disks onto tape
  - Largest market for disks → well developed
  - A small market for tapes → under developed
- Advantages eroded by disk technology developments
- Currently, ATA disks are cheaper than tapes
- Makes better sense to replicate data over multiple sites, which depends on advances in disk and network
  - e.g, RAID, remote mirroring

Fallacy: Disk Scheduling

- OS are the best place to schedule disk accesses
- Higher-level interface like ATA and SCSI offer logical block addresses to the host OS.
  - The best OS can do to improve performance is to sort the logical block address into increasing order.
- The disks know the actual mapping of the logical block addresses onto the physical locations,
  - It can reduce the rotational latency by rescheduling.
    - Map to physical track, cylinder, sector locations
    - Also, blocks are cached by the drive
  - OS is unaware of physical locations
    - Reordering can reduce performance
    - Depending on placement and caching

OS vs disk schedule accesses



Operation	Starting LBA	Length
Read	724	8
Read	100	16
Read	9987	1
Read	26	128

Host-ordered queue
Drive-ordered queue

OS Scheduled: address sorting

Operation	Starting LBA	Length
Read	26	128
Read	100	16
Read	724	8
Read	9987	1

Example showing OS versus disk schedule accesses, labeled host-ordered versus drive-ordered. The former takes three revolutions to complete the four reads, while the latter completes them in just three-fourths of a revolution (from Anderson [2003]).

Pitfall: Peak Performance

- Peak I/O rates are nearly impossible to achieve
  - Usually, some other system component limits performance
  - E.g., transfers to memory over a bus
    - Collision with DRAM refresh
    - Arbitration contention with other bus masters
  - E.g., PCI bus: peak bandwidth ~133 MB/sec
    - In practice, max 80MB/sec sustainable

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
- I/O benchmarks
  - TPC, SPECSFS, SPECWeb
- RAID
  - Improves performance and dependability

Homework: chapter 6

- Due before starting the class on Nov. 29
- Exercise 6.3
- Exercise 6.6
- Exercise 6.11
- Exercise 6.15
- Exercise 6.16