Chapter 6

Memory 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
Hierarchical Bus Architecture

Chapter 6 — Storage and Other I/O Topics
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

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

Chapter 6 — Storage and Other I/O Topics
Typical x86 PC I/O System

Chapter 6 — Storage and Other I/O Topics
# Intel & AMD I/O chip sets

## Chapter 6 — Storage and Other I/O Topics

<table>
<thead>
<tr>
<th></th>
<th>Intel 5000P chip set</th>
<th>Intel 975X chip set</th>
<th>AMD 580X CrossFire</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target segment</strong></td>
<td>Server</td>
<td>Performance PC</td>
<td>Server/Performance PC</td>
</tr>
<tr>
<td><strong>Front Side Bus (64 bit)</strong></td>
<td>1066/1333 MHz</td>
<td>800/1066 MHz</td>
<td>—</td>
</tr>
</tbody>
</table>

### Memory controller hub ("north bridge")

<table>
<thead>
<tr>
<th></th>
<th>Intel 5000P MCH</th>
<th>975X MCH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pins</strong></td>
<td>1432</td>
<td>1202</td>
</tr>
<tr>
<td><strong>Memory type, speed</strong></td>
<td>DDR2 FBDIMM 667/533</td>
<td>DDR2 800/667/533</td>
</tr>
<tr>
<td><strong>Memory buses, widths</strong></td>
<td>4 × 72</td>
<td>1 × 72</td>
</tr>
<tr>
<td><strong>Number of DIMMs, DRAM/DIMM</strong></td>
<td>16, 1 GB/2 GB/4 GB</td>
<td>4, 1 GB/2 GB</td>
</tr>
<tr>
<td><strong>Maximum memory capacity</strong></td>
<td>64 GB</td>
<td>8 GB</td>
</tr>
<tr>
<td><strong>Memory error correction available?</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>PCIe/External Graphics Interface</strong></td>
<td>1 PCIe x16 or 2 PCIe x</td>
<td>1 PCIe x16 or 2 PCIe x8</td>
</tr>
<tr>
<td><strong>South bridge interface</strong></td>
<td>PCIe x8, ESI</td>
<td>PCIe x8</td>
</tr>
</tbody>
</table>

### I/O controller hub ("south bridge")

<table>
<thead>
<tr>
<th></th>
<th>6321 ESB</th>
<th>ICH7</th>
<th>580X CrossFire</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package size, pins</strong></td>
<td>1284</td>
<td>652</td>
<td>549</td>
</tr>
<tr>
<td><strong>PCI-bus: width, speed</strong></td>
<td>Two 64-bit, 133 MHz</td>
<td>32-bit, 33 MHz, 6 masters</td>
<td>—</td>
</tr>
<tr>
<td><strong>PCI Express ports</strong></td>
<td>Three PCIe x4</td>
<td>—</td>
<td>Two PCIe x16, Four PCI x1</td>
</tr>
<tr>
<td><strong>Ethernet MAC controller, interface</strong></td>
<td>—</td>
<td>1000/100/10 Mbit</td>
<td>—</td>
</tr>
<tr>
<td><strong>USB 2.0 ports, controllers</strong></td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td><strong>ATA ports, speed</strong></td>
<td>One 100</td>
<td>Two 100</td>
<td>One 133</td>
</tr>
<tr>
<td><strong>Serial ATA ports</strong></td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td><strong>AC-97 audio controller, interface</strong></td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>I/O management</strong></td>
<td>SMbus 2.0, GPIO</td>
<td>SMbus 2.0, GPIO</td>
<td>ASF 2.0, GPIO</td>
</tr>
</tbody>
</table>
ARM Advanced Microcontroller Bus Architecture (AMBA)

- On-chip interconnect specification for SoC
- Promotes re-use by defining a common backbone for SoC modules using standard bus architectures
  - AHB – Advanced High-performance Bus (system backbone)
    - High-performance, high clock freq. modules
    - Processors to on-chip memory, off-chip memory interfaces
  - APB – Advanced Peripheral Bus
    - Low-power peripherals
    - Reduced interface complexity
  - ASB – Advanced System Bus
    - High performance alternate to AHB
  - AXI – Advanced eXtensible Interface
  - ACE – AXI Coherency Extension
  - ATB – Advanced Trace Bus
Example AMBA System

Joe Bungo (ARM):
CPU Design Concept to SoC

Chapter 6 — Storage and Other I/O Topics
NXP LPC2292 Microcontroller

ARM buses

Chapter 6 — Storage and Other I/O Topics
## ARM Cortex-A9 System IP

### Interconnect SoC components

<table>
<thead>
<tr>
<th>Description</th>
<th>AMBA Bus</th>
<th>System IP Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced AMBA 3 Interconnect IP</td>
<td>AXI</td>
<td>NIC-301, PL301</td>
</tr>
<tr>
<td>DMA Controller</td>
<td>AXI</td>
<td>DMA-330, PL330</td>
</tr>
<tr>
<td>Level 2 Cache Controller</td>
<td>AXI</td>
<td>L2C-310, PL310</td>
</tr>
<tr>
<td>Dynamic Memory Controller</td>
<td>AXI</td>
<td>DMC-340, PL340</td>
</tr>
<tr>
<td>DDR2 Dynamic Memory Controller</td>
<td>AXI</td>
<td>DMC-342</td>
</tr>
<tr>
<td>Static Memory Controller</td>
<td>AXI</td>
<td>SMC-35x, PL35x</td>
</tr>
<tr>
<td>TrustZone Address Space Controller</td>
<td>AXI</td>
<td>PL380</td>
</tr>
<tr>
<td>CoreSight™ Design Kit</td>
<td>ATB</td>
<td>CDK-11</td>
</tr>
</tbody>
</table>

Chapter 6 — Storage and Other I/O Topics
“CoreLink” = interconnect + memory controllers for Cortex/Mali
Clock provides synchronization.

R/W is true when reading (R/W’ is false when reading).

Address is a-bit bundle of address lines.

Data is n-bit bundle of data lines.

Data ready signals when n-bit data is ready.
Bus read and write
State diagrams for bus read

- Get data → Done
- See ack
- Wait → CPU
- Send data → Ack
- Ack → Wait → device
- Release ack

start
Bus wait state

Clock

R/W

Address enable

Address

Data ready

Data

Time

Wait state

Computers as Components 2nd ed. © 2008 Wayne Wolf
Bus burst read

Clock
R/W
Burst
Address enable
Address
Data ready

Data 1  Data 2  Data 3  Data 4

Time
Asynchronous Bus Handshaking Protocol

Output (read) data from memory to an I/O device

I/O device signals a request by raising ReadReq and putting the addr on the data lines

1. Memory sees ReadReq, reads addr from data lines, and raises Ack
2. I/O device sees Ack and releases the ReadReq and data lines
3. Memory sees ReadReq go low and drops Ack
4. When memory has data ready, it places it on data lines and raises DataRdy
5. I/O device sees DataRdy, reads the data from data lines, and raises Ack
6. Memory sees Ack, releases the data lines, and drops DataRdy
7. I/O device sees DataRdy go low and drops Ack
Memory device organization

- Size.
  - Address width.
    - \( n = r + c \)
  - DRAM:
    - mux \( r \) & \( c \) bits
- Aspect ratio.
  - Data width \( d \)

Memory array

Memory “organization” = \( 2^n \times d \) (from system designer's perspective)
Typical generic SRAM

- Often have separate OE' and WE' instead of one R/W' signal.
- Multi-byte Data bus devices usually have byte-select signals.
512K x 16 SRAM (on uCdragon board)
Generic SRAM timing

- CE'
- R/W'
- Adrs
- Data

From SRAM
From CPU

read
write
time
ISSI IS61LV51216 SRAM read cycle
# ISSI IS61LV51216 SRAM Timing

## READ CYCLE SWITCHING CHARACTERISTICS

(Over Operating Range)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>-8 Min.</th>
<th>Max.</th>
<th>-10 Min.</th>
<th>Max.</th>
<th>-12 Min.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tRC</td>
<td>Read Cycle Time</td>
<td>8</td>
<td></td>
<td>10</td>
<td></td>
<td>12</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tAA</td>
<td>Address Access Time</td>
<td>—</td>
<td>8</td>
<td>—</td>
<td>10</td>
<td>—</td>
<td>12</td>
<td>ns</td>
</tr>
<tr>
<td>tOHA</td>
<td>Output Hold Time</td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tACE</td>
<td>CE Access Time</td>
<td>—</td>
<td>8</td>
<td>—</td>
<td>10</td>
<td>—</td>
<td>12</td>
<td>ns</td>
</tr>
<tr>
<td>tDOE</td>
<td>OE Access Time</td>
<td>—</td>
<td>3.5</td>
<td>—</td>
<td>4</td>
<td>—</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>tHZOE(2)</td>
<td>OE to High-Z Output</td>
<td>—</td>
<td>3</td>
<td>—</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>tLZOE(2)</td>
<td>OE to Low-Z Output</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tHZCE(2)</td>
<td>CE to High-Z Output</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>ns</td>
</tr>
<tr>
<td>tLZCE(2)</td>
<td>CE to Low-Z Output</td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tBA</td>
<td>LB, UB Access Time</td>
<td>—</td>
<td>3.5</td>
<td>—</td>
<td>4</td>
<td>—</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>tHZB(2)</td>
<td>LB, UB to High-Z Output</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>ns</td>
</tr>
<tr>
<td>tLZB(2)</td>
<td>LB, UB to Low-Z Output</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tPU</td>
<td>Power Up Time</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tPD</td>
<td>Power Down Time</td>
<td>—</td>
<td>8</td>
<td>—</td>
<td>10</td>
<td>—</td>
<td>12</td>
<td>ns</td>
</tr>
</tbody>
</table>
Design example #1

- Find the bandwidth of a synchronous bus
  - Clock period $T = 50$ns
  - 1 cycle required to xmit address/data
  - Bus width = 32 bits
  - Memory access time = 200ns

  \[
  \text{Address + Memory-Read + Send_Data} = 50\text{ns} + 200\text{ns} + 50\text{ns} = 300\text{ns}
  \]

  \[
  \text{BW} = 4 \text{ Bytes}/300\text{ns} = 13.3 \text{ Mbytes/sec}
  \]

- Burst transfer 4 words from synchronous DRAM, at one clock each for words 2-3-4

  \[
  50\text{ns} + 200\text{ns} + 50\text{ns} + (3 \times 50\text{ns}) = 450\text{ns}
  \]

  \[
  \text{BW} = 16 \text{ Bytes}/450\text{ns} = 35.6 \text{ Mbytes/sec}
  \]
Design example #2

- Asynchronous bus
  - 40ns to complete each “handshake” (HS)
  - Memory access time = 200ns
  - 32-bit data bus

\[(\text{addr}) \quad (\text{mem-read}) \quad (\text{data xfer})\]

\[
\begin{align*}
\text{HS-1} + & \quad \text{HS-2-3-4*} + \quad \text{HS-5-6-7} \\
40\text{ns} + & \quad 200\text{ns} + \quad 3 \times 40\text{ns} = 360\text{ns} \\
\text{BW} = & \quad 4 \text{ Bytes/360ns} = 11.1 \text{ Mbytes/sec}
\end{align*}
\]

- Memory read (200ns) concurrent with 3 handshakes
- (3 x 40ns), so memory read time dominates
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
  - 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
  - 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
Bus mastership

- **Bus master** controls operations on the bus.
  - CPU is default bus master.
- Other devices may request bus mastership.
  - Separate set of handshaking lines.
  - CPU can’t use bus when it is not master.
- Situations for multiple bus masters:
  - DMA data transfers
  - Multiple CPUs with shared memory
    - One CPU might be graphics/network processor
**DMA organization**

- *Direct memory access (DMA)* performs data transfers without executing instructions.
  - CPU configures transfer in DMA controller
  - DMA controller fetches & writes data.
- DMA controller is a separate unit.
  - CPU is the default bus master
DMA operation

- CPU sets DMA registers for start address, length.
- DMA controller has to acquire control of the bus
  - Bus request – ask for control of bus
  - Bus grant – acknowledgement that request granted
- Once DMA is bus master, it transfers automatically.
  - May run continuously until complete.
  - May use every n\textsuperscript{th} bus cycle.
  - CPU cannot use bus while DMA controller is master
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
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O
DMA/VM Interaction

- OS uses virtual addresses for memory
  - DMA blocks may not be contiguous in physical memory
- Should DMA use virtual addresses?
  - Would require controller to do translation
- If DMA uses physical addresses
  - May need to break transfers into page-sized chunks
  - Or chain multiple transfers
  - Or allocate contiguous physical pages for DMA
Measuring I/O Performance

- 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
Transaction Processing Benchmarks

- Transactions
  - Small data accesses to a DBMS
  - Interested in I/O rate, not data rate

- 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
File System & Web 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
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

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU time</th>
<th>I/O time</th>
<th>Elapsed time</th>
<th>% I/O time</th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td>90s</td>
<td>10s</td>
<td>100s</td>
<td>10%</td>
</tr>
<tr>
<td>+2</td>
<td>45s</td>
<td>10s</td>
<td>55s</td>
<td>18%</td>
</tr>
<tr>
<td>+4</td>
<td>23s</td>
<td>10s</td>
<td>33s</td>
<td>31%</td>
</tr>
<tr>
<td>+6</td>
<td>11s</td>
<td>10s</td>
<td>21s</td>
<td>47%</td>
</tr>
</tbody>
</table>
I/O System Design

- Satisfying latency requirements
  - For time-critical operations
  - If system is unloaded
    - Add up latency of components

- Maximizing throughput
  - Find “weakest link” (lowest-bandwidth component)
  - Configure to operate at its maximum bandwidth
  - Balance remaining components in the system

- If system is loaded, simple analysis is insufficient
  - Need to use queuing models or simulation
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
  - Multiples of 1.75” (1U) high
Rack-Mounted Servers

Sun Fire x4150 1U server

- 2 Redundant power Supplies
- 3 PCI Express Slots
- System Status LEDs
- Management NIC
- Management Serial
- 2 USB Ports
- 4 Gigabit NICs
- Video
Sun Fire x4150 1U server

4 cores each

16 x 4GB = 64GB DRAM

Chapter 6 — Storage and Other I/O Topics
I/O System Design Example

What I/O rate can be sustained?
- For random reads, and for sequential reads

Given a Sun Fire x4150 system with
- Workload: 64KB disk reads
  - Each I/O op requires 200,000 user-code instructions and 100,000 OS instructions
- Each CPU: $10^9$ instructions/sec
- FSB: 10.6 GB/sec peak
- DRAM DDR2 667MHz: 5.336 GB/sec
- PCI-E 8× bus: $8 \times 250$MB/sec = 2GB/sec
- Disks: 15,000 rpm, 2.9ms avg. seek time, 112MB/sec transfer rate

Chapter 6 — Storage and Other I/O Topics
Design Example (cont)

- I/O rate for CPUs
  - Per core: \( \frac{10^9}{(100,000 + 200,000)} = 3,333 \)
  - 8 cores: 26,667 ops/sec

- Random reads, I/O rate for disks
  - Assume actual seek time is average/4
  - Time/op = seek + latency + transfer
    \[ = \frac{2.9\text{ms}}{4} + \frac{4\text{ms}}{2} + \frac{64\text{KB}}{112\text{MB/s}} = 3.3\text{ms} \]
  - 303 ops/sec per disk, 2424 ops/sec for 8 disks

- Sequential reads
  - 112MB/s / 64KB = 1750 ops/sec per disk
  - 14,000 ops/sec for 8 disks
Design Example (cont)

- **PCI-E I/O rate** (RAID -> North Bridge)
  - 2GB/sec / 64KB = 31,250 ops/sec
- **DRAM I/O rate** (MCB -> DRAM)
  - 5.336 GB/sec / 64KB = 83,375 ops/sec
- **FSB I/O rate** (North Bridge -> CPU)
  - Assume we can sustain half the peak rate
  - 5.3 GB/sec / 64KB = 81,540 ops/sec per FSB
  - 163,080 ops/sec for 2 FSBs
- **Weakest link: disks**
  - 2424 ops/sec random, 14,000 ops/sec sequential
  - Other components have ample headroom to accommodate these rates
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
Pitfall: Offloading to I/O Processors

- Overhead of managing I/O processor request may dominate
  - Quicker to do small operation on the CPU
  - But I/O architecture may prevent that
- I/O processor may be slower
  - Since it’s supposed to be simpler
- Making it faster makes it into a major system component
  - Might need its own coprocessors!
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