Introduction to Input and Output

• The I/O subsystem provides the mechanism for communication between the CPU and the outside world (I/O devices).

• Design factors:
  – I/O device characteristics (input, output, storage, etc.).
  – I/O Connection Structure (degree of separation from memory operations).
  – I/O interface (the utilization of dedicated I/O and bus controllers).
  – Types of buses (processor-memory vs. I/O buses).
  – I/O data transfer or synchronization method (programmed I/O, interrupt-driven, DMA).
Typical CPU-Memory and I/O Bus Interface

[Diagram showing the flow of data between CPU, memory, cache, I/O controllers, disks, graphics output, and network.]
Impact of I/O on System Performance

• CPU Performance: Improvement of 60% per year.

• I/O Sub-System Performance: Limited by mechanical delays (disk I/O). Improvement less than 10% per year (IO rate per sec or MB per sec).

• From Amdahl's Law: overall system speed-up is limited by the slowest component:

   If I/O is 10% of current processing time:
   • Increasing CPU performance by 10 times
     ⇒ Only 5 times system performance increase (50% loss in performance)
   • Increasing CPU performance by 100 times
     ⇒ Only 10 times system performance increase (90% loss of performance)

• The I/O system performance bottleneck diminishes the benefit of faster CPUs on overall system performance.
I/O Device Characteristics

• I/O devices are characterized according to:

  – Behavior:
    • Input (read once).
    • Output (write only, cannot be read).
    • Storage (can be reread and usually rewritten).

  – Partner: Either a human or a machine at the other end of the I/O device.

  – Data rate: The peak rate at which data can be transferred between the I/O device and main memory or CPU.
# The Diversity of I/O Devices

<table>
<thead>
<tr>
<th>Device</th>
<th>Behavior</th>
<th>Partner</th>
<th>Data Rate (KB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>Input</td>
<td>Human</td>
<td>0.01</td>
</tr>
<tr>
<td>Mouse</td>
<td>Input</td>
<td>Human</td>
<td>0.02</td>
</tr>
<tr>
<td>Line Printer</td>
<td>Output</td>
<td>Human</td>
<td>1.00</td>
</tr>
<tr>
<td>Floppy disk</td>
<td>Storage</td>
<td>Machine</td>
<td>50.00</td>
</tr>
<tr>
<td>Laser Printer</td>
<td>Output</td>
<td>Human</td>
<td>100.00</td>
</tr>
<tr>
<td>Optical Disk</td>
<td>Storage</td>
<td>Machine</td>
<td>500.00</td>
</tr>
<tr>
<td>Magnetic Disk</td>
<td>Storage</td>
<td>Machine</td>
<td>5,000.00</td>
</tr>
<tr>
<td>Network-LAN</td>
<td>Input or Output</td>
<td>Machine</td>
<td>20 – 1,000.00</td>
</tr>
<tr>
<td>Graphics Display</td>
<td>Output</td>
<td>Human</td>
<td>30,000.00</td>
</tr>
</tbody>
</table>
I/O System Performance

- I/O System performance depends on many aspects of the system ("limited by weakest link in the chain"):
  - The CPU
  - The memory system:
    - Internal and external caches.
    - Main Memory.
  - The underlying interconnection (buses).
  - The I/O controller or hardware interface.
  - The I/O device itself.
  - I/O Data Transfer Method.
  - The speed of the I/O software (Operating System).
  - The efficiency of the software’s use of the I/O devices.

- Two common performance metrics:
  - Throughput: I/O bandwidth.
  - Response time: Latency.
I/O Performance: Simple Producer-Server Model

- **Throughput:**
  - The number of tasks completed by the server in unit time.
  - In order to get the highest possible throughput:
    - The server should never be idle.
    - The queue should never be empty.

- **Response time:**
  - Begins when a task is placed in the queue
  - Ends when it is completed by the server
  - In order to minimize the response time:
    - The queue should be empty.
    - The server will be idle at times.
Throughput Versus Response Time

Response Time (ms)

Percentage of maximum throughput
I/O Performance:
Throughput Enhancement

• In general throughput can be improved by:
  – Throwing more hardware at the problem.
  – Reduces load-related latency.

• Response time is much harder to reduce.
Magnetic Disks

Characteristics:

- Diameter: 2.5in - 5.25in
- Rotational speed: 3,600RPM-10,000 RPM
- Tracks per surface.
- Sectors per track: Outer tracks contain more sectors.
- Recording or Areal Density: Tracks/in X Bits/in
- Cost Per Megabyte.
- Seek Time: The time needed to move the read/write head arm.
  Reported values: Minimum, Maximum, Average.
- Rotation Latency or Delay:
  The time for the requested sector to be under the read/write head.
- Transfer time: The time needed to transfer a sector of bits.
- Type of controller/interface: SCSI, EIDE
- Disk Controller delay or time.
- Average time to access a sector of data =
  average seek time + average rotational delay + transfer time +
  disk controller overhead + Queueing Delay
# Magnetic Disk Examples

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>IBM 3090</th>
<th>IBM UltraStar</th>
<th>Integral 1820</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk diameter (inches)</td>
<td>10.88</td>
<td>3.50</td>
<td>1.80</td>
</tr>
<tr>
<td>Formatted data capacity (MB)</td>
<td>22,700</td>
<td>4,300</td>
<td>21</td>
</tr>
<tr>
<td>MTTF (hours)</td>
<td>50,000</td>
<td>1,000,000</td>
<td>100,000</td>
</tr>
<tr>
<td>Number of arms/box</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Rotation speed (RPM)</td>
<td>3,600</td>
<td>7,200</td>
<td>3,800</td>
</tr>
<tr>
<td>Transfer rate (MB/sec)</td>
<td>4.2</td>
<td>9-12</td>
<td>1.9</td>
</tr>
<tr>
<td>Power/box (watts)</td>
<td>2,900</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>MB/watt</td>
<td>8</td>
<td>102</td>
<td>10.5</td>
</tr>
<tr>
<td>Volume (cubic feet)</td>
<td>97</td>
<td>0.13</td>
<td>0.02</td>
</tr>
<tr>
<td>MB/cubic feet</td>
<td>234</td>
<td>33000</td>
<td>1050</td>
</tr>
</tbody>
</table>
• Disk Access Time = Seek time + Rotational Latency + Transfer time
  + Controller Time + Queueing Delay

• Estimating Queue Length:
  – Utilization = U = Request Rate / Service Rate
  – Mean Queue Length = U / (1 - U)
  – As Request Rate → Service Rate
    • Mean Queue Length → Infinity
Disk Access Time Example

- Given the following Disk Parameters:
  - Transfer size is 8K bytes
  - Advertised average seek is 12 ms
  - Disk spins at 7200 RPM
  - Transfer rate is 4 MB/sec

- Controller overhead is 2 ms

- Assume that the disk is idle, so no queuing delay exist.

- What is Average Disk Access Time for a 512-byte Sector?
  - Ave. seek + ave. rot delay + transfer time + controller overhead + Queueing Delay
  - 12 ms + 0.5/(7200 RPM/60) + 8 KB/4 MB/s + 2 ms + 0
  - 12 + 4.15 + 2 + 2 + 0 = 20 ms

- Advertised seek time assumes no locality: typically 1/4 to 1/3 advertised seek time: 20 ms => 12 ms
I/O Connection Structure

Different computer system architectures use different degrees of separation between I/O data transmission and memory transmissions.

- **Isolated I/O:** Separate memory and I/O buses.
  - A set of I/O device address, data and control lines form a separate I/O bus.
  - Special input and output instructions are used to handle I/O operations.

- **Shared I/O:**
  - Address and data wires are shared between I/O and memory buses.
  - Different control lines for I/O control.
  - Different I/O instructions.

- **Memory-mapped I/O:**
  - Shared address, data, and control lines for memory and I/O.
  - Data transfer to/from the CPU is standardized.
  - Common in modern processor design; reduces CPU chip connections.
  - A range of memory addresses is reserved for I/O registers.
  - I/O registers read/written using standard load/store instructions.
I/O Interface

I/O Interface, controller or I/O bus adapter:

- Specific to each type of I/O device.
- To the CPU, and I/O device, it consists of a set of control and data registers within the I/O address space.
- On the I/O device side, it forms a localized I/O bus which can be shared by several I/O devices.
- Handles I/O details such as:
  - Assembling bits into words,
  - Low-level error detection and correction.
  - Accepting or providing words in word-sized I/O registers.
  - Presents a uniform interface to the CPU regardless of I/O device.
Types of Buses

- Processor-Memory Bus (sometimes also called Backplane Bus):
  - Offers very high-speed and low latency.
  - Matched to the memory system to maximize memory-processor bandwidth.
  - Usually design-specific, though some designs use standard backplane buses.

- I/O buses (sometimes called a channel):
  - Follow bus standards.
  - Usually formed by I/O interface adapters to handle many types of connected I/O devices.
  - Wide range in the data bandwidth and latency
  - Not usually interfaced directly to memory but use a processor-memory or backplane bus.
  - Examples: Sun’s SBus, Intel’s PCI, SCSI.
Processor-Memory, I/O Bus Organization
# Main Bus Characteristics

<table>
<thead>
<tr>
<th>Option</th>
<th>High performance</th>
<th>Low cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus width</td>
<td>Separate address &amp; data lines</td>
<td>Multiplex address &amp; data lines</td>
</tr>
<tr>
<td>Data width</td>
<td>Wider is faster (e.g., 32 bits)</td>
<td>Narrower is cheaper (e.g., 8 bits)</td>
</tr>
<tr>
<td>Transfer size</td>
<td>Multiple words has less bus overhead</td>
<td>Single-word transfer is simpler</td>
</tr>
<tr>
<td>Bus masters</td>
<td>Multiple (requires arbitration)</td>
<td>Single master (no arbitration)</td>
</tr>
<tr>
<td>Split</td>
<td>Yes, separate Request and Reply packets gets higher bandwidth (needs multiple masters)</td>
<td>No, continuous transaction? connection is cheaper and has lower latency</td>
</tr>
<tr>
<td>Clocking</td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
</tbody>
</table>
Synchronous Vs. Asynchronous Buses

• Synchronous Bus:
  – A clock is included in the control lines.
  – A fixed communication protocol relative to the clock is used. (memory-processor communication).

• Asynchronous Bus:
  – Not clocked.
  – A handshaking protocol is used: A series of steps in which the sender and receiver proceed to the next step when both agree using an additional set of control lines.
  – A device requesting a word from memory may use the lines:
    • ReadReq: Read request from memory. Address put on the data lines at the same time.
    • DataRdy: Indicated that a data word is now ready.
    • Ack: Used to acknowledge the ReadReq or DataRdy signal of the other party.
Asynchronous Bus Handshaking Protocol Steps

Details in Figure 8.10 page 661
Typical Synchronous Bus Read Transaction

- The read begins when the read signal is deasserted
- Data not ready until the wait signal is deasserted
Split-transaction Bus

- Used when multiple bus masters are present,
- Also known as a pipelined or a packet-switched bus
- The bus is available to other bus masters while a memory operation is in progress
  ⇒ Higher bus bandwidth, but also higher bus latency

A read transaction is tagged and broken into:
  - A read request-transaction containing the address
  - A memory-reply transaction that contains the data
  ⇒ address on the bus refers to a later memory access
Performance Analysis of Two Bus Schemes

Example in textbook pages 665-667
Obtaining Access to the Bus: Bus Arbitration

Bus arbitration decides which device (bus master) gets the use of the bus next. Several schemes exist:

- **A single bus master:**
  - All bus requests are controlled by the processor.

- **Daisy chain arbitration:**
  - A bus grant line runs through the devices from the highest priority to lowest (priority determined by the position on the bus).
  - A high-priority device intercepts the bus grant signal, not allowing a low-priority device to see it (VME bus).

- **Centralized, parallel arbitration:**
  - Multiple request lines for each device.
  - A centralized arbiter chooses a requesting device and notifies it that it is now the bus master.
Obtaining Access to the Bus: Bus Arbitration

• Distributed arbitration by self-selection:
  – Use multiple request lines for each device
  – Each device requesting the bus places a code indicating its identity on the bus.
  – The requesting devices determine the highest priority device to control the bus.
  – Requires more lines for request signals (Apple NuBus).

• Distributed arbitration by collision detection:
  – Each device independently request the bus.
  – Multiple simultaneous requests result in a collision.
  – The collision is detected and a scheme to decide among the colliding requests is used (Ethernet).
Bus Transactions with a Single Master

Details in Figure 8.12 page 668
Daisy Chain Bus Arbitration

Details in Figure 8.13 page 670
# I/O Bus Examples

<table>
<thead>
<tr>
<th>Bus</th>
<th>SBus</th>
<th>TurboChannel</th>
<th>MicroChannel</th>
<th>PCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>Sun</td>
<td>DEC</td>
<td>IBM</td>
<td>Intel</td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>16-25</td>
<td>12.5-25</td>
<td>async</td>
<td>33</td>
</tr>
<tr>
<td>Addressing</td>
<td>Virtual</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Data Sizes (bits)</td>
<td>8,16,32</td>
<td>8,16,24,32</td>
<td>8,16,24,32,64</td>
<td>8,16,24,32,64</td>
</tr>
<tr>
<td>Master</td>
<td>Multi</td>
<td>Single</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td>32 bit read (MB/s)</td>
<td>33</td>
<td>25</td>
<td>20</td>
<td>33</td>
</tr>
<tr>
<td>Peak (MB/s)</td>
<td>89</td>
<td>84</td>
<td>75</td>
<td>111 (222)</td>
</tr>
<tr>
<td>Max Power (W)</td>
<td>16</td>
<td>26</td>
<td>13</td>
<td>25</td>
</tr>
</tbody>
</table>
## CPU-Memory Bus Examples

<table>
<thead>
<tr>
<th>Bus</th>
<th>Summit</th>
<th>Challenge</th>
<th>XDBus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>HP</td>
<td>SGI</td>
<td>Sun</td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>60</td>
<td>48</td>
<td>66</td>
</tr>
<tr>
<td>Split transaction?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Address lines</td>
<td>48</td>
<td>40</td>
<td>??</td>
</tr>
<tr>
<td>Data lines</td>
<td>128</td>
<td>256</td>
<td>144 (parity)</td>
</tr>
<tr>
<td>Data Sizes (bits)</td>
<td>512</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>Clocks/transfer</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Peak (MB/s)</td>
<td>960</td>
<td>1200</td>
<td>1056</td>
</tr>
<tr>
<td>Master</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td>Addressing</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Slots</td>
<td>16</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Busses/system</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Length</td>
<td>13 inches</td>
<td>12 inches</td>
<td>17 inches</td>
</tr>
</tbody>
</table>
SCSI: Small Computer System Interface

- Clock rate: 5 MHz / 10 MHz (fast) / 20 MHz (ultra).
- Width: $n = 8$ bits / 16 bits (wide); up to $n - 1$ devices to communicate on a bus or “string”.
- Devices can be slave (“target”) or master (“initiator”).
- SCSI protocol: A series of “phases”, during which specific actions are taken by the controller and the SCSI disks and devices.
  - Bus Free: No device is currently accessing the bus
  - Arbitration: When the SCSI bus goes free, multiple devices may request (arbitrate for) the bus; fixed priority by address
  - Selection: Informs the target that it will participate (Reselection if disconnected)
  - Command: The initiator reads the SCSI command bytes from host memory and sends them to the target
  - Data Transfer: data in or out, initiator: target
  - Message Phase: message in or out, initiator: target (identify, save/restore data pointer, disconnect, command complete)
  - Status Phase: target, just before command complete.
SCSI "Bus": Channel Architecture

peer-to-peer protocols
initiator/target
linear byte streams
disconnect/reconnect

Command Setup
Arbitration
Selection
Message Out (Identify)
Command

Disconnect to seek/ll buffer
Message In (Disconnect)
- - Bus Free - -
Arbitration
Reselection
Message In (Identify)

If no disconnect is needed

Data Transfer
Data In

Disconnect to ll buffer
Message In (Save Data Ptr)
Message In (Disconnect)
- - Bus Free - -
Arbitration
Reselection
Message In (Identify)
Message In (Restore Data Ptr)

Completion

Command Completion
Status
Message In (Command Complete)
P1394 High-Speed Serial Bus (firewire)

- A digital interface: No need to convert digital data into analog signals and tolerate a loss of data integrity,
- Physically small: A thin serial cable can replace larger and more expensive interfaces,
- Easy to use: No need for terminators, device IDs, or elaborate setup,
- Hot pluggable - users can add or remove 1394 devices with the bus active,
- Inexpensive - priced for consumer products,
- Scalable architecture: May mix 100, 200, and 400 Mbps devices on a bus,
- Flexible topology: support of daisy chaining and branching for true peer-to-peer communication,
- Fast: Even multimedia data can be guaranteed its bandwidth for just-in-time delivery, and
- Non-proprietary.
- Mixed asynchronous and isochronous traffic.
Firewire Operations

Packet Frame = 125 µsecs

- Fixed frame is divided into preallocated isochronous slots + best effort asynchronous slot
- Each slot has packet containing “ID” command and data
- Example: digital video camera can expect to send one 64 byte packet every 125 µs:
  \[80 \times 1024 \times 64 = 5\text{MB/s}\]
I/O Data Transfer Methods

- **Programmed I/O (PIO): Polling**
  - The I/O device puts its status information in a status register.
  - The processor must periodically check the status register.
  - The processor is totally in control and does all the work.
  - Very wasteful of processor time.

- **Interrupt-Driven I/O:**
  - An interrupt line from the I/O device to the CPU is used to generate an I/O interrupt indicating that the I/O device needs CPU attention.
  - The interrupting device places its identity in an interrupt vector.
  - Once an I/O interrupt is detected the current instruction is completed and an I/O interrupt handling routine is executed to service the device.
Polling: Programmed I/O

- **Advantage:**
  - Simple: the processor is totally in control and does all the work.
- **Disadvantage:**
  - Polling overhead can consume a lot of CPU time.

**busy wait loop**
not an efficient way to use the CPU unless the device is very fast!

but checks for I/O completion can be dispersed among computation intensive code
Polling Example

Example in textbook pages 676-678
Interrupt-Driven Data Transfer

- Advantage:
  - User program progress is only halted during actual transfer.

- Disadvantage, special hardware is needed to:
  - Cause an interrupt (I/O device).
  - Detect an interrupt (processor).
  - Save the proper states to resume after the interrupt (processor).
Interrupt-driven I/O Example

Example in textbook pages 679-680
I/O Data Transfer Methods

Direct Memory Access (DMA):

- Implemented with a specialized controller that transfers data between an I/O device and memory independent of the processor.
- The DMA controller becomes the bus master and directs reads and writes between itself and memory.
- Interrupts are still used only on completion of the transfer or when an error occurs.
- DMA transfer steps:
  - The CPU sets up DMA by supplying device identity, operation, memory address of source and destination of data, the number of bytes to be transferred.
  - The DMA controller starts the operation. When the data is available it transfers the data, including generating memory addresses for data to be transferred.
  - Once the DMA transfer is complete, the controller interrupts the processor, which determines whether the entire operation is complete.
Direct Memory Access (DMA) Example

Example in textbook pages 681-682