Chapter 6

Storage and Other I/O Topics
Introduction

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

- I/O bus connections
Interconnecting Components

- **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
- **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

<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>Intended use</td>
<td>External</td>
<td>External</td>
<td>Internal</td>
<td>Internal</td>
<td>External</td>
</tr>
<tr>
<td>Devices per channel</td>
<td>63</td>
<td>127</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Data width</td>
<td>4</td>
<td>2</td>
<td>2/lane</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Peak bandwidth</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>Hot pluggable</td>
<td>Yes</td>
<td>Yes</td>
<td>Depends</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Max length</td>
<td>4.5m</td>
<td>5m</td>
<td>0.5m</td>
<td>1m</td>
<td>8m</td>
</tr>
<tr>
<td>Standard</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>

---

*IEEE 1394, USB Implementers Forum, PCI-SIG, SATA-IO, INCITS TC T10*
Typical x86 PC I/O System

- **Intel Xeon 5300 processor**
- **Front Side Bus (1333 MHz, 10.5 GB/sec)**
- **Main memory DIMMs**
  - **FB DDR2 667 (5.3 GB/sec)**
- **Memory controller hub (north bridge) 5000P**
  - **PCIe x16 (or 2 PCIe x8) (4 GB/sec)**
  - **ESI (2 GB/sec)**
  - **PCle x8 (2 GB/sec)**
- **I/O controller hub (south bridge) Enterprise South Bridge 2**
  - **PCIe x4 (1 GB/sec)**
  - **PCIe x4 (1 GB/sec)**
  - **PCI-X bus (1 GB/sec)**
  - **PCI-X bus (1 GB/sec)**
- **Disk**
  - **Serial ATA (300 MB/sec)**
  - **LPC (1 MB/sec)**
  - **USB 2.0 (60 MB/sec)**
- **Disk**
- **Keyboard, mouse, ...**
- **Parallel ATA (100 MB/sec)**
- **CD/DVD**
I/O Approaches

• Memory mapped I/O
  ▪ Device 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

- 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 (vector or cause register)

• Priority interrupts
  ▪ Devices needing more urgent attention get higher priority
  ▪ Can block 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/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