Chapter 8

A Typical collection of I/O devices
Interfacing Processors and Peripherals

- I/O Design affected by many factors (expandability, resilience)
- Performance:
  - access latency
  - throughput
  - connection between devices and the system
  - the memory hierarchy
  - the operating system
- A variety of different users (e.g., banks, supercomputers, engineers)

I/O Devices

- Very diverse devices
  - behavior (i.e., input vs. output)
  - partner (who is at the other end?)
  - data rate

<table>
<thead>
<tr>
<th>Device</th>
<th>Behavior</th>
<th>Partner</th>
<th>Data rate (Mb/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>input</td>
<td>human</td>
<td>0.0001</td>
</tr>
<tr>
<td>Mouse</td>
<td>input</td>
<td>human</td>
<td>0.0038</td>
</tr>
<tr>
<td>Voice input</td>
<td>input</td>
<td>human</td>
<td>0.2630</td>
</tr>
<tr>
<td>Scanner</td>
<td>input</td>
<td>human</td>
<td>3.2</td>
</tr>
<tr>
<td>Voice output</td>
<td>output</td>
<td>human</td>
<td>0.264</td>
</tr>
<tr>
<td>Laser printer</td>
<td>output</td>
<td>human</td>
<td>3.2</td>
</tr>
<tr>
<td>Graphics display</td>
<td>output</td>
<td>human</td>
<td>800-8000</td>
</tr>
<tr>
<td>Modem</td>
<td>input or output</td>
<td>machine</td>
<td>0.016-0.064</td>
</tr>
<tr>
<td>Network/LAN</td>
<td>input or output</td>
<td>machine</td>
<td>100-1000</td>
</tr>
<tr>
<td>Optical disk</td>
<td>storage</td>
<td>machine</td>
<td>80</td>
</tr>
<tr>
<td>Magnetic tape</td>
<td>storage</td>
<td>machine</td>
<td>32</td>
</tr>
<tr>
<td>Magnetic disk</td>
<td>storage</td>
<td>machine</td>
<td>240-2560</td>
</tr>
</tbody>
</table>
I/O Example: Disk Drives

To access data:
- seek: position head over the proper track (3 to 14 ms. avg.)
- rotational latency: wait for desired sector (.5 / RPM)
- transfer: grab the data (one or more sectors) 30 to 80 MB/sec

Buses

- Buses connect I/O devices to processors and memory
- A bus is a shared communication link which uses one set of wires to connect multiple systems
- Advantages
  - Shared link, therefore cost effective
- Disadvantage
  - Shared link can become a communication bottleneck
- Each bus has a set of *data lines* and a set of *control lines*
Using a Bus for Input/Output

- Each bus transaction has two parts
  - Sending the address
  - Receiving/Sending the data
- Input transaction
  - Inputs data from a device to memory
- Output transaction
  - Outputting data from memory to a device

The three steps of an output transaction

1. Memory
2. Control lines
3. Data lines
4. Disks

a.

b.

c.
An input transaction

Types of buses
## Types of buses

- **Processor-Memory bus**
  - Short
  - High-speed
  - Matched to the memory system
- **I/O buses**
  - Lengthy
  - Can have many different devices attached to them
  - Wide range of data transfer speeds
- **Backplane buses**
  - Can have processor, memory, I/O devices coexisting on a single bus
  - Balance demands of processor-memory communication with demands of device memory communication

## Synchronous and Asynchronous Buses

- **Synchronous**
  - Driven by a clock
  - Clock input to control lines
  - Protocol for communication is relative to this clock
  - Example: processor-memory communication
    - Processor sends address and read command in first clock cycle
    - Memory sends data on fifth clock cycle
    - **Protocol is predetermined and driven by the clock**
  - Disadvantage
    - Every device on bus must be able to run at same clock rate
    - Synchronous buses have to be short to avoid clock skew
Synchronous & Asynchronous Buses

- Asynchronous Buses
  - Not clocked
  - Can be lengthened
  - Use a handshaking protocol
- Synchronous buses typically faster than asynchronous

Handshaking Protocol for an Asynchronous Bus

1. When memory sees ReadReq line, it reads the address from the Data bus and raises ACK to indicate it has been seen
2. I/O device sees ACK and releases ReadReq and Data lines
3. Memory sees ReadReq is low and drops the ACK signal
4. This step starts when memory has the data ready. It places data on the Data line and raises DataRdy
5. I/O device sees DataRdy, reads the data, and then raises ACK
6. The memory sees ACK, and drops DataRdy and releases the data lines
7. I/O device sees DataRdy go low and drops the ACK line
Obtaining Access to the Bus

- How is the bus reserved by one of the devices that wants to use it to communicate?
- Single Bus Master: the CPU
  - Advantage: simple
  - Disadvantage: processor involved in every bus transaction
- Multiple bus masters
- Need Bus Arbitration:
  - daisy chain arbitration (not very fair)
  - centralized arbitration (requires an arbiter), e.g., PCI
  - self selection, e.g., NuBus used in old Macintosh
  - collision detection, e.g., Ethernet

I/O Bus Standards

- Today we have two dominant bus standards:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Firewire (1994)</th>
<th>USB 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus type</td>
<td>I/O</td>
<td>I/O</td>
</tr>
<tr>
<td>Basic-data bus width (signals)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Clocking</td>
<td>asynchronous</td>
<td>asynchronous</td>
</tr>
<tr>
<td>Theoretical peak bandwidth</td>
<td>50 MB/sec (Firewire 400) or 100 MB/sec (Firewire 800)</td>
<td>0.2 MB/sec (low speed), 1.5 MB/sec (full speed), or 50 MB/sec (high-speed)</td>
</tr>
<tr>
<td>Hot pluggable</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Maximum number of devices</td>
<td>63</td>
<td>127</td>
</tr>
<tr>
<td>Maximum bus length (cooperative)</td>
<td>4.5 meters</td>
<td>5 meters</td>
</tr>
<tr>
<td>Standard name</td>
<td>IEEE 1394, 1344b</td>
<td>USB implementers Forum</td>
</tr>
</tbody>
</table>

FIGURE 8.3 Key characteristics of two dominant I/O bus standards.
Buses and Networks of the Pentium 4

- I/O Options

![Diagram of Pentium 4 buses and networks]

<table>
<thead>
<tr>
<th>Buses and Networks</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel ATA</td>
<td>(100 MB/sec)</td>
</tr>
<tr>
<td>Serial ATA</td>
<td>(150 MB/sec)</td>
</tr>
<tr>
<td>PCI bus</td>
<td>(132 MB/sec)</td>
</tr>
<tr>
<td>AGP 8X</td>
<td>(2.1 GB/sec)</td>
</tr>
<tr>
<td>Memory controller</td>
<td>1 Gbit Ethernet</td>
</tr>
<tr>
<td>Main memory (DIMMs)</td>
<td>DDR 400 (3.2 GB/sec)</td>
</tr>
<tr>
<td>Memory controller</td>
<td>DDR 400 (3.2 GB/sec)</td>
</tr>
<tr>
<td>Memory controller</td>
<td>Serial ATA</td>
</tr>
<tr>
<td>Memory controller</td>
<td>AC97 (1 MB/sec)</td>
</tr>
<tr>
<td>Memory controller</td>
<td>USB 2.0 (60 MB/sec)</td>
</tr>
<tr>
<td>Memory controller</td>
<td>AUX 10/100 Mbit Ethernet</td>
</tr>
</tbody>
</table>

Interfacing I/O devices to the Memory, Processor, and Operating System

- How is a user I/O request transformed into a device command and communicated to the device?
  - E.g., file read/write, mouse movement, keyboard stroke

- How is data actually transferred to or from a memory location?

- What is the role of the operating system?
Operating System Responsibilities wrt I/O system

- Characteristics of I/O devices
  - Shared by multiple programs
  - Interrupt driven
  - Low-level control is complex
- OS functions
  - Must provide protection
    - E.g., must not allow file owned by one user to be deleted by another user
  - Must provide abstractions for accessing device
    - E.g., file abstraction for a collection of blocks on disk
  - Must handle interrupts
  - Must try to provide “fairness” in accessing I/O devices
  - Must try and manage I/O devices so that throughput is maximized

Communication between I/O devices and the OS

- To perform its functions wrt I/O system, the operating system must be able to communicate with I/O devices and to prevent user programs from accessing the I/O devices directly
- Three types of communication
  - OS must be able to give commands to I/O devices
  - A device must be able to notify the OS when it has completed a command or if there is an error
  - Data must be transferred between the I/O device and memory
Giving commands to I/O devices

- CPU must be able to address the device and to supply one or more commands
- Two methods for addressing the device
  - Memory-mapped I/O
    - Portions of a program's address space are assigned to I/O devices
    - Reads and writes to these addresses are interpreted as commands to the device
    - These memory addresses are not directly accessible to user programs
  - Special I/O instructions
    - I/O instructions can specify both the device number and the command word (or the location of the command word in memory)
    - I/O instructions can only be executed by the operating system

Communication with the Processor

- Two methods
  - Polling
    - Device status bits are periodically checked to see if it is time for the next I/O operation
  - Interrupt-driven I/O
    - Device delivers interrupt to the CPU when it requires attention
    - Interrupts are like exceptions except that they are not associated with any instruction
    - CPU can check before starting a new instruction if an interrupt has been delivered
Polling vs Interrupt-driven I/O

Assume that the number of clock cycles for a polling operation is 100. For a processor executes at 50 MHz, what is the overhead of polling?

1. For a mouse that is polled 30 times per second?
2. For a floppy disk that transfers data to the processor in 16-bit units and has a data transfer rate of 50 KB/second?
3. For a hard disk transferring data in 1 word chunks at 2 MB/sec?

For the mouse
- clock cycles used per second for polling = 30 x 100 = 300
- Fraction of processor cycles used for polling = \( \frac{300}{50 \times 10^6} \) = 0.006%

For the floppy drive
- Number of polling operations per second (if we don’t want to lose data) = \( \frac{50 \text{ KB/sec}}{2 \text{ bytes/access}} \) = 25 K polling accesses per second
- Clock cycles for polling = 25 K x 100 = 25 x 1024 x 100 = 25.6 x 10^5 clock cycles per second
- Fraction of CPU cycles = \( \frac{25.6 \times 10^5}{50 \times 10^6} \) = 5%

For the hard disk
- Rate of polling = \( \frac{(2 \text{ MB/sec})}{4 \text{ bytes per access}} \) = 500 K polling accesses per second
- Clock cycles = 500 K x 100 = 51.2 x 10^6
- Fraction of CPU cycles = \( \frac{51.2 \times 10^6}{50 \times 10^6} \) = 100% !!!!
Polling vs Interrupt-driven I/O  cont’d

- Suppose overhead of interrupt handling is 100 clock cycles. How much overhead when floppy disk is active?

Rate of interrupts = \( \frac{50 \text{ KB/sec}}{2 \text{ bytes/interrupt}} \)

- 25 K interrupts per second

Clock cycles for handling interrupts

- \( 25 \text{ K} \times 100 = 25 \times 1024 \times 100 \)
- \( 25.6 \times 10^5 \) clock cycles per second

Fraction of CPU cycles = \( \frac{25.6 \times 10^5}{50 \times 10^6} = 5\% \)

**The difference from polling is that 5% of the CPU cycles per second are used for handling interrupts only if the floppy is busy.**

At other times, overhead is 0%. For polling, the overhead is always 5%.

Transferring data between device and memory

- Two methods
  - Interrupt-driven I/O
    - Processor is involved in data transfer
    - Problem: 100% overhead in the case of the hard disk example
  - Direct Memory Access (DMA)
    - Data is transferred directly from the device to memory (or vice versa)
    - Processor is involved only in
      1. Initiating the DMA transfer
      2. Handling interrupt at the end of DMA transfer
DMA

- Implemented with special controller that transfers data between memory and I/O device independent of the processor
- Three steps in DMA transfers
  1. Processor sets up the DMA transfer by supplying identity of device, operation to perform, memory address that is source or destination of data, number of bytes to be transferred
  2. DMA controller starts the operation (arbitrates for the bus, supplies address, reads or writes data), until the entire block is transferred
  3. DMA controller interrupts the processor, which then takes the necessary actions

Hard disk DMA example

Find overhead for using DMA for data transfer from a hard disk.
Assume initial DMA setup = 1000 cycles
Interrupt handling on DMA completion = 500 cycles
Average size of data transfer = 4 KB

Each DMA transfer takes \((4 \text{ KB})/(2 \text{ MB/sec}) = 2 \times 10^{-3}\) seconds
CPU cycles used for DMA transfer = 1000 + 500 = 1500
Total CPU cycles during DMA transfer = \((50 \times 10^6) \times (2 \times 10^{-3}) = 100 \times 10^3\)
Fraction of CPU cycles used for DMA = \(1500 / 100 \times 10^3 = 1.5\%\)