IC220 Set #19: Storage and I/O (Chapter 8)

ADMIN

- Reading – Chapter 8
  - Including RAID (8.2) but don’t stress memorizing the levels
  - Can skip 8.8

I/O

- Important but neglected
  - “The difficulties in assessing and designing I/O systems have often relegated I/O to second class status”
  - “courses in every aspect of computing, from programming to computer architecture often ignore I/O or give it scanty coverage”
  - “textbooks leave the subject to near the end, making it easier for students and instructors to skip it!”

- GUILTY!
  - we won’t be looking at I/O in much detail
  - be sure and read Chapter 8 carefully
  - Later – IC322: Computer Networks
(A) I/O Overview

- Can characterize devices based on:
  1. behavior
  2. partner (who is at the other end?)
  3. data rate

- Performance factors:
  - access latency
  - throughput
  - connection between devices and the system
  - the memory hierarchy
  - the operating system

- Other issues:
  - Expandability, dependability

(B) Connecting the Processor, Memory, and other Devices

Two general strategies:

1. Bus: ______ communication link
   - Advantages:
   - Disadvantages:

2. Point to Point Network: _____________ links
   - Use switches to enable multiple connections
   - Advantages:
   - Disadvantages:

(B) Bus Basics – Part 1

- Types of buses:
  - Process-memory
    - Short, high speed, fixed device types
    - custom design
  - I/O
    - lengthy, different devices
    - Standards-based e.g., USB, Firewire
    - Connect to proc-memory bus rather than directly to processor
  - Only one pair of devices (sender & receiver) may use bus at a time
    - Bus _____________ decides who gets the bus next based on
      some _____________ strategy
    - May incorporate priority, round-robin aspects
  - Have two types of signals:
    - “Data” – data or address
    - Control
I/O Bus Standards

• Today we have two dominant I/O bus standards:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Firewire (1.3M)</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>Serial data bus width (bits)</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 400/1200)</td>
<td>0.2 MB/sec (low speed), 1.6 MB/sec (full speed), or 60 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>66</td>
<td>127</td>
</tr>
<tr>
<td>Maximum bus length</td>
<td>3.5 meters</td>
<td>5 meters</td>
</tr>
<tr>
<td>Standard name</td>
<td>IEEE 1394, 1596</td>
<td>USB implementors forum</td>
</tr>
</tbody>
</table>

Figure 8.9 Key characteristics of two dominant I/O bus standards.

(B) Bus Basics – Part 2

• Clocking scheme:
  1. ____________________
     - Use a clock, signals change only on clock edge
     + Fast and small
     - All devices must operate at same rate
     - Requires bus to be short (due to clock skew)
  2. ____________________
     - No clock, instead use “handshaking”
     + Longer buses possible
     + Accommodate wide range of device
     - More complex control

(C) Processor-to-device Communication

How does CPU send information to a device?
1. Special I/O instructions
   x86: inb / outb

   How to control access to I/O device?

2. Use normal load/instructions to special addresses
   Called ____________________
   Load/store put onto bus
   Memory ignores them (outside its range)
   Address may encode both device ID and a command

   How to control access to I/O device?

Handshaking example – CPU read from memory

1. CPU requests read
2. Memory acknowledges, CPU deasserts request
3. Memory sees change, deasserts Ack
4. Memory provides data, asserts DataRdy
5. CPU grabs data, asserts Ack
6. Memory sees Ack, deasserts DataRdy
7. CPU sees change, deasserts Ack
(C) Device-to-processor communication

How does device get data to the processor?
1. CPU periodically checks to see if device is ready: _________________
   • CPU sends request, keep checking if done
   • Or just checks for new info (mouse, network)

2. Device forces action by the processor when needed: _________________
   • Like an unscheduled procedure call
   • Same as “exception” mechanism that handles TLB misses, divide by zero, etc.

3. DMA:
   • Device sends data directly to memory w/o CPU’s involvement
   • Interrupts CPU when transfer is complete

(D) I/O’s impact on performance

- Total time = CPU time + I/O time
- Suppose our program is 90% CPU time, 10% I/O. If we improve CPU performance by 10x, but leave I/O unchanged, what will the new performance be?
  - Old time = 100 seconds
  - New time =

(D) Measuring I/O Performance

- Latency?
- Throughput?
- Throughput with maximum latency?
- Transaction processing benchmarks
  - TPC-C
  - TPC-H
  - TPC-W

- File system / Web benchmarks
  - “Make” benchmark
  - SPECSFS
  - SPECWeb
**(E) 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

**RAID**

- RAID 0 – “striping”, no redundancy
- RAID 1 – mirrored
- RAID 4 – Block-interleaved parity
- RAID 5 – Distributed Block-interleaved Parity

**RAID**

- Idea: lots of cheap, smaller disks
- Small size and cost makes easier to add redundancy
- Multiple disks increases read/write bandwidth
RAID

RAID 10 – Striped mirrors

- Key point – still need to do other backups (e.g. to tape)
  - Provides protection from limited number of disk failures
  - No protection from human failures!

Fallacies and Pitfalls

- Fallacy: the rated mean time to failure of disks is 1,200,000 hours, so disks practically never fail.

- Fallacy: magnetic disk storage is on its last legs, will be replaced.

- Fallacy: A 100 MB/sec bus can transfer 100 MB/sec.

- Pitfall: Moving functions from the CPU to the I/O processor, expecting to improve performance without analysis.