I/O Management

Intro

Chapter 5
I/O Devices

• There exists a large variety of I/O devices:
  – Many of them with different properties
  – They seem to require different interfaces to manipulate and manage them
    • We don’t want a new interface for every device
    • Diverse, but similar interfaces leads to code duplication

• Challenge:
  – Uniform and efficient approach to I/O
Categories of I/O Devices (by usage)

- **Human interface**
  - Used to communicate with the user
  - Printers, Video Display, Keyboard, Mouse

- **Machine interface**
  - Used to communicate with electronic equipment
  - Disk and tape drives, Sensors, Controllers, Actuators

- **Communication**
  - Used to communicate with remote devices
  - Ethernet, Modems, Wireless
I/O Device Handling

• Data rate
  – May be differences of several orders of magnitude between the data transfer rates

  – Example: Assume 1000 cycles/byte I/O
    • Keyboard needs 10 KHz processor to keep up
    • Gigabit Ethernet needs 100 GHz processor…..
## Sample Data Rates

<table>
<thead>
<tr>
<th>Device</th>
<th>Data rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>10 bytes/sec</td>
</tr>
<tr>
<td>Mouse</td>
<td>100 bytes/sec</td>
</tr>
<tr>
<td>56K modem</td>
<td>7 KB/sec</td>
</tr>
<tr>
<td>Telephone channel</td>
<td>8 KB/sec</td>
</tr>
<tr>
<td>Dual ISDN lines</td>
<td>16 KB/sec</td>
</tr>
<tr>
<td>Laser printer</td>
<td>100 KB/sec</td>
</tr>
<tr>
<td>Scanner</td>
<td>400 KB/sec</td>
</tr>
<tr>
<td>Classic Ethernet</td>
<td>1.25 MB/sec</td>
</tr>
<tr>
<td>USB (Universal Serial Bus)</td>
<td>1.5 MB/sec</td>
</tr>
<tr>
<td>Digital camcorder</td>
<td>4 MB/sec</td>
</tr>
<tr>
<td>IDE disk</td>
<td>5 MB/sec</td>
</tr>
<tr>
<td>40x CD-ROM</td>
<td>6 MB/sec</td>
</tr>
<tr>
<td>Fast Ethernet</td>
<td>12.5 MB/sec</td>
</tr>
<tr>
<td>ISA bus</td>
<td>16.7 MB/sec</td>
</tr>
<tr>
<td>EIDE (ATA-2) disk</td>
<td>16.7 MB/sec</td>
</tr>
<tr>
<td>FireWire (IEEE 1394)</td>
<td>50 MB/sec</td>
</tr>
<tr>
<td>XGA Monitor</td>
<td>60 MB/sec</td>
</tr>
<tr>
<td>SONET OC-12 network</td>
<td>78 MB/sec</td>
</tr>
<tr>
<td>SCSI Ultra 2 disk</td>
<td>80 MB/sec</td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>125 MB/sec</td>
</tr>
<tr>
<td>Utlrium tape</td>
<td>320 MB/sec</td>
</tr>
<tr>
<td>PCI bus</td>
<td>528 MB/sec</td>
</tr>
<tr>
<td>Sun Gigaplane XB backplane</td>
<td>20 GB/sec</td>
</tr>
</tbody>
</table>
I/O Device Handling Considerations

- Complexity of control
- Unit of transfer
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- Data representation
  - Encoding schemes
- Error conditions
  - Devices respond to errors differently
    - `lp0`: printer on fire!
  - Expected error rate also differs
I/O Device Handling Considerations

• Layering
  – Need to be both general and specific, e.g.
  – Devices that are the same, but aren’t the same
    • Hard-disk, USB disk, RAM disk
  – Interaction of layers
    • Swap partition and data on same disk
    • Two mice
  – Priority
    • Keyboard, disk, network
Accessing I/O Controllers

a) Separate I/O and memory space
   – I/O controller registers appear as I/O ports
   – Accessed with special I/O instructions

b) Memory-mapped I/O
   – Controller registers appear as memory
   – Use normal load/store instructions to access

c) Hybrid
   – x86 has both ports and memory mapped I/O
   – *Linux Device Drivers*; Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman
Bus Architectures

(a) A single-bus architecture
(b) A dual-bus memory architecture
Intel IXP420

The diagram illustrates the architecture of the Intel IXP420 processor. It includes the following components:

- **Ethernet NPE A Ethernet MAC**
- **Ethernet NPE B Ethernet MAC**
- **Intel XScale® Core**
  - 266/400/533 MHz
  - 32 KB Data Cache
  - 32 KB Instruction Cache
  - 2 KB Mini-Data Cache
- **SDRAM Controller 8 - 256 MB**
- **Queue Manager 8 KB SRAM**
- **North AHB Arbiter**
- **South AHB Arbiter**
- **AHB/APB Bridge**
- **133.32 MHz x 32 bits North Advance High-Performance Bus**
- **Queue Status Bus**
- **133.32 MHz x 32 bits South Advance High-Performance Bus**
- **AMI-0**
- **AMI-1**
- **56.66 MHz Advanced Peripheral Bus**
- **16 GPIO**
- **Test Logic Unit**
- **PMU (AHB)**
- **USB Device V1.1**
- **GPIO Controller**
- **Interrupt Controller**
- **UART 921Kbaud**
- **UART 921Kbaud**

The diagram shows the connections and interfaces between these components, indicating how they interact within the IXP420 processor.
Interrupts Revisited

- Devices connected to an *Interrupt Controller* via lines on an I/O bus (e.g. PCI)
- Interrupt Controller signals interrupt to CPU and is eventually acknowledged.
- Exact details are architecture specific (APIC is most common).
Programmed I/O

- Also called *polling*, or *busy waiting*
- I/O module (controller) performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete
  - Wastes CPU cycles
Interrupt-Driven I/O

- Processor is interrupted when I/O module (controller) ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor
Direct Memory Access

• Transfers a block of data directly to or from memory
• An interrupt is sent when the task is complete
• The processor is only involved at the beginning and end of the transfer
DMA Considerations

✓ Reduces number of interrupts
  – Less (expensive) context switches

✗ Requires contiguous regions
  – Copying
  – Scatter-gather

• Synchronous/Asynchronous
• Shared bus must be arbitrated
DMA

• Cycle stealing is used to transfer data on the system bus
  – The instruction cycle is suspended so data can be transferred
  – The CPU pauses one bus cycle
    • CPU Cache can hopefully avoid such pauses by hide DMA bus transactions
  – Cycle stealing causes the CPU to execute more slowly
    • Still more efficient than CPU doing transfer itself

Very Simplified Model of Cycle Stealing

[Diagram showing system bus cycles, CPU cycles, and DMA cycles]
DMA

• Commonly *burst-mode* is used
  – CPU uses several consecutive cycles to load entire cache line
  – DMA writes (or reads) a similar sized burst
  – Reason: More efficient (less cycles overall) to transfer a sequence of words than a word at a time.
    • No bus arbitration, read/write setup, or addressing cycles required after first transfer.

• Number of required busy cycles can be cut by
  – Path between DMA module and I/O module that does not include the system bus
The Process to Perform DMA Transfer

5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion

1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X

IDE disk controller

3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
Evolution of the I/O Function

• Processor directly controls a peripheral device
  – Example: CPU controls a flip-flop to implement a serial line
Evolution of the I/O Function

- Controller or I/O module is added
  - Processor uses programmed I/O without interrupts
  - Processor does not need to handle details of external devices
  - Example: A Universal Asynchronous Receiver Transmitter
    - CPU simply reads and writes bytes to I/O controller
    - I/O controller responsible for managing the signaling
Evolution of the I/O Function

- Controller or I/O module with interrupts
  - Processor does not spend time waiting for an I/O operation to be performed
Evolution of the I/O Function

• Direct Memory Access
  – Blocks of data are moved into memory without involving the processor
  – Processor involved at beginning and end only
Evolution of the I/O Function

• I/O module has a separate processor
  – Example: SCSI controller
    • Controller CPU executes SCSI program code out of main memory
Evolution of the I/O Function

• I/O processor
  – I/O module has its own local memory, internal bus, etc.
  – It's a computer in its own right
  – Example: Myrinet 10 gigabit NIC