CS415 Module 8 Part A - I/O Hardware

Athens State University

Outline

Contents

1 What Exactly Are These Things Doing?

Categories of I/O Devices

- Three main categories of I/O devices

  Human readable Things that communicate with the user: printers, terminals, video displays, keyboard, mouse

  Machine readable Suitable for communicating with electronic equipment: disk drives, USB memory sticks, sensors, controllers

  Communications Suitable for communicating with remote devices: modems, digital line drivers, network interfaces

Devices differ in many ways

Data rate Multiple orders of magnitude differences

Application Use of the device influences the software

Complexity of control Effect on OS is filtered by the complexity of the device

Unit of Transfer Data may be transferred as a stream of bytes or characters

Data Representation Different data encoding schemes are used by different devices

Error conditions The nature of errors, and how they are addressed, differs from device to device

2 How The Operating System Interfaces With Hardware

Ways To Interface With Hardware

Programmed I/O Processor issues an I/O command on behalf of the process to an I/O unit and process busy waits for operation to complete

Interrupt-driven I/O The processor issues an I/O command on behalf of a process:

- If non-blocking, processor continues to execute instructions from the process that issued the command
- If blocking, the OS will block current process and schedule another

Direct Memory Access (DMA) A DMA hardware unit controls the exchange of data between main memory and an I/O module

Another View of The H/W Inteface

	No Interrupts	**Use of interrupts**
I/O to Memory thru CPU	Programmed I/O	Interrupt-driven I/O
Direct I/O transfer		DMA

How the I/O Function Has Evolved

1. Processor directly controls a peripheral device
2. A controller or I/O module is added
3. Same as before, but managed through interrupts rather than directly
4. The I/O module is given direct control of memory via DMA
5. The I/O module becomes a separate procesor, with specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right

2.1 Direct Memory Access

Direct Memory Access

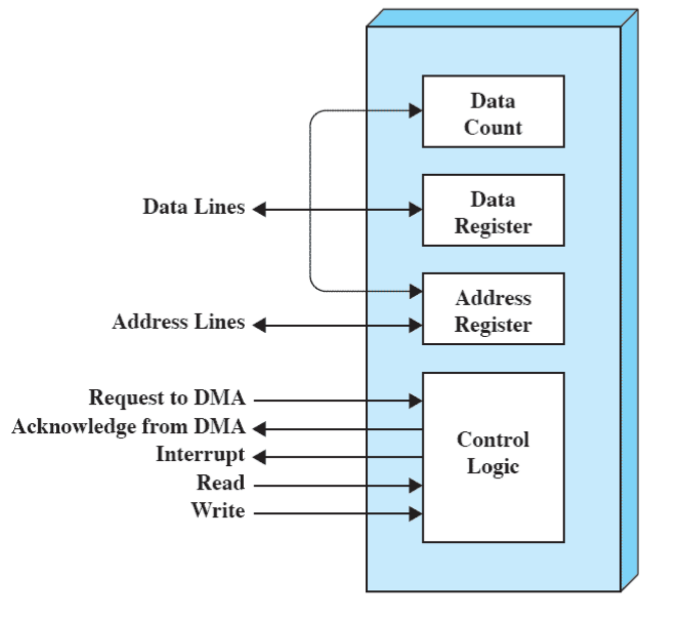
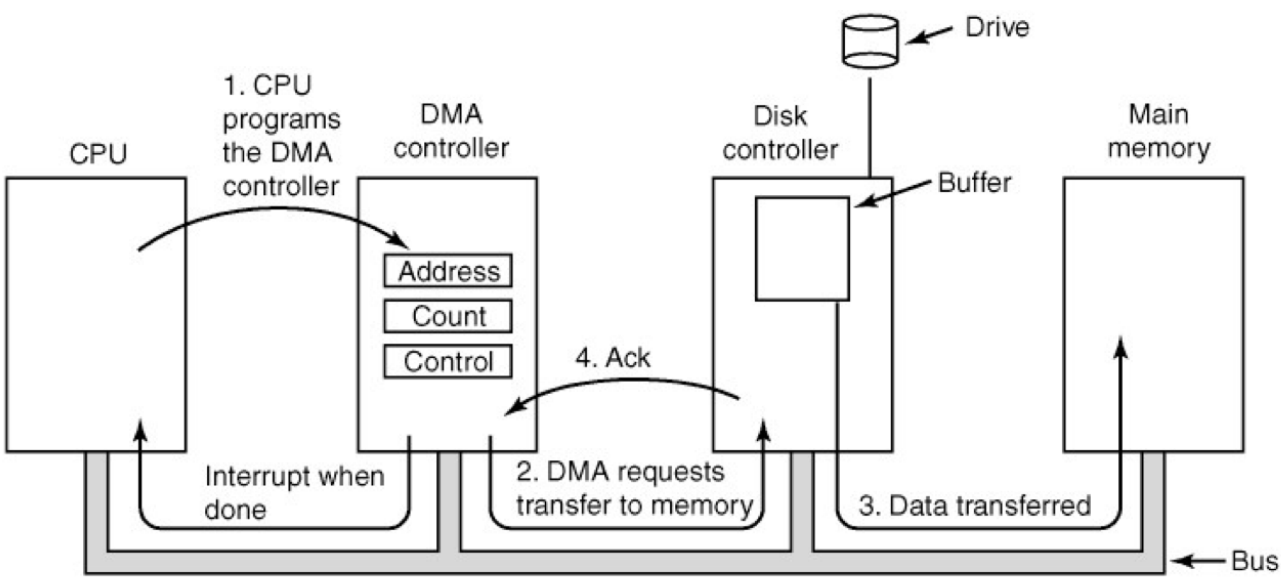


Figure 11.2 Typical DMA Block Diagram

- The DMA controller is a co-processor with the CPU
- Shares the system bus with the CPU

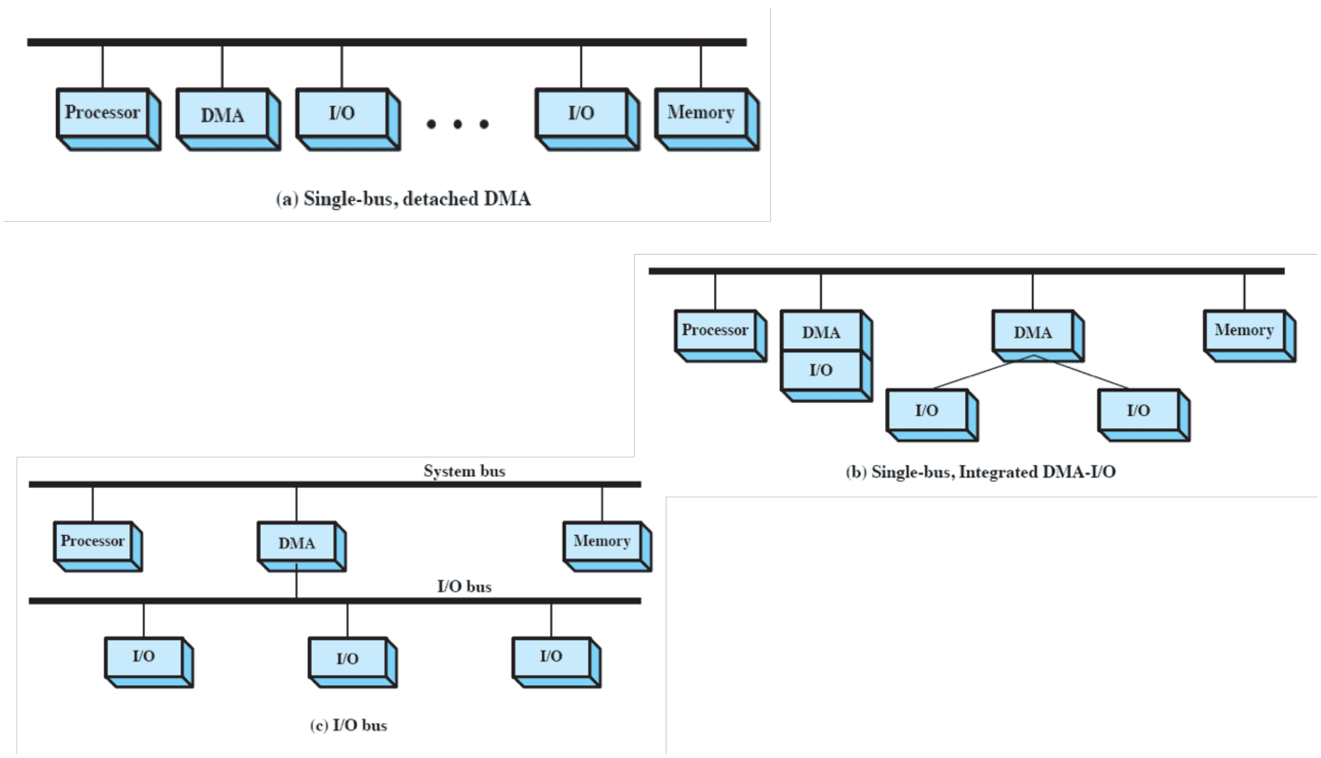
How DMA Operates



How DMA operates

- Whether a read or write is requested, using the read or write control line between the processor and the DMA module
- The address of the I/O device involved, communicated on the data lines
- The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register
- The number of words to be read or written, again communicated via the data lines and stored in the data count register

Alternative DMA Organizations



(a) Single-bus, detached DMA

(b) Single-bus, Integrated DMA-I/O

(c) I/O bus

3 Buffering

Buffering

Perform input transfers in advance of requests being made and perform output transfers some time after the request is made

Block-oriented device

- Store information in blocks that are of fixed size
- Transfer one block at time
- Reference data by block number

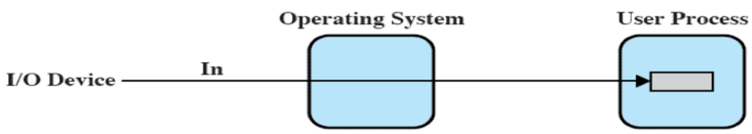
- Examples: Disks, SSDs, USB drives

Stream-oriented device

- Transfers data in and out as a stream of bytes
- No block structure
- Examples:Terminals, printers, network hardware

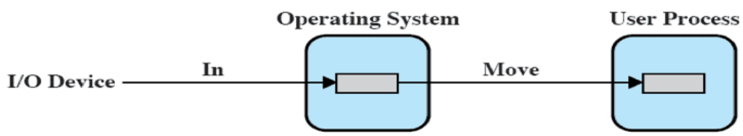
Buffering

No buffer



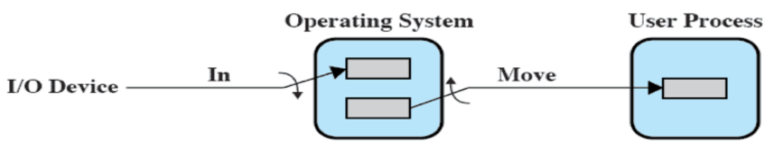
(a) No buffering

Single buffer



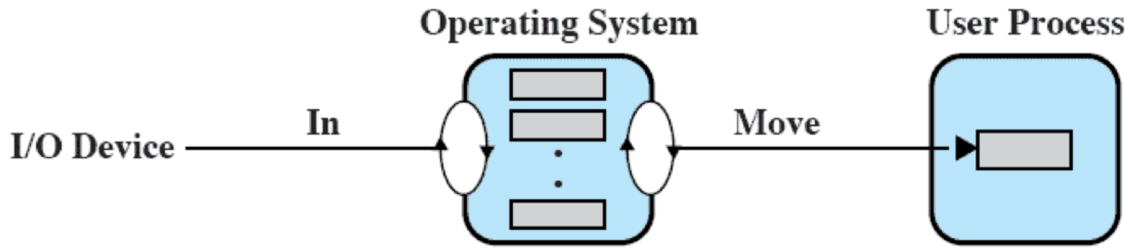
(b) Single buffering

Double buffer



(c) Double buffering

Circular buffers



(d) Circular buffering

- To or more buffers are used
- Each buffer is treated as single unit in this scheme
- Used when the I/O op. must keep up with the process

Why use buffering?

- Smoothes out peaks in I/O demand
  - At some demand will cause all buffers to become full and advantage is lost
- Improves efficiency when variety of I/O and process activities needing service

4 Implications On Operating System Design

Design Objectives

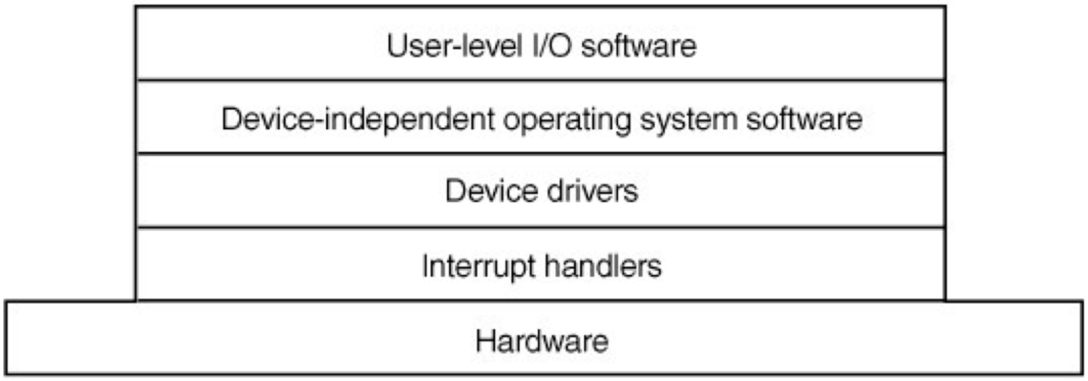
Efficiency

- Major effort in I/O design
- Important because I/O often a perf. bottleneck
- Most I/O devices extremely slow compared with memory and CPU
- The area that has received the most attention is disk I/O

Generality

- Wish to handle all devices uniformly
- Apples to both how process views I/O and how OS manages I/O devices
- Diversity of devices makes this difficult
- Use a hierarchical and modular approach to the design of I/O function

The Hierarchical Layer Cake



The Hierarchical Layer Cake

- Functions of the OS need to be separated according to their complexity, characteristic time scale, and level of abstraction
- Leads to the OS being designed using a "Seven-layer Bean-dip Architecture" pattern
- Each layer performs a related subset of the functions required by the operating systems
- Layers need to be defined so that changes in one layer do require changes in other layers (remember policy vs. mechanism?)