Input/Output Systems

Dr. B. R. Bhowmik
Dept. of CSE
NIT Karnataka

Input/Output Architectures

- The computer system's I/O architecture is its interface to the outside world.
- This architecture provides a systematic means of controlling interaction with the outside world and provides the operating system with the information it needs to manage I/O activity effectively.
- The are three principal I/O techniques
  a) **programmed I/O -** I/O occurs under the direct and continuous control of the program requesting the I/O operation.
  b) **interrupt-driven I/O** - a program issues an I/O command and then continues to execute, until it is interrupted by the I/O hardware to signal the end of the I/O operation.
  c) **direct memory access (DMA)**- a specialized I/O processor takes over control of an I/O operation to move a large block of data.

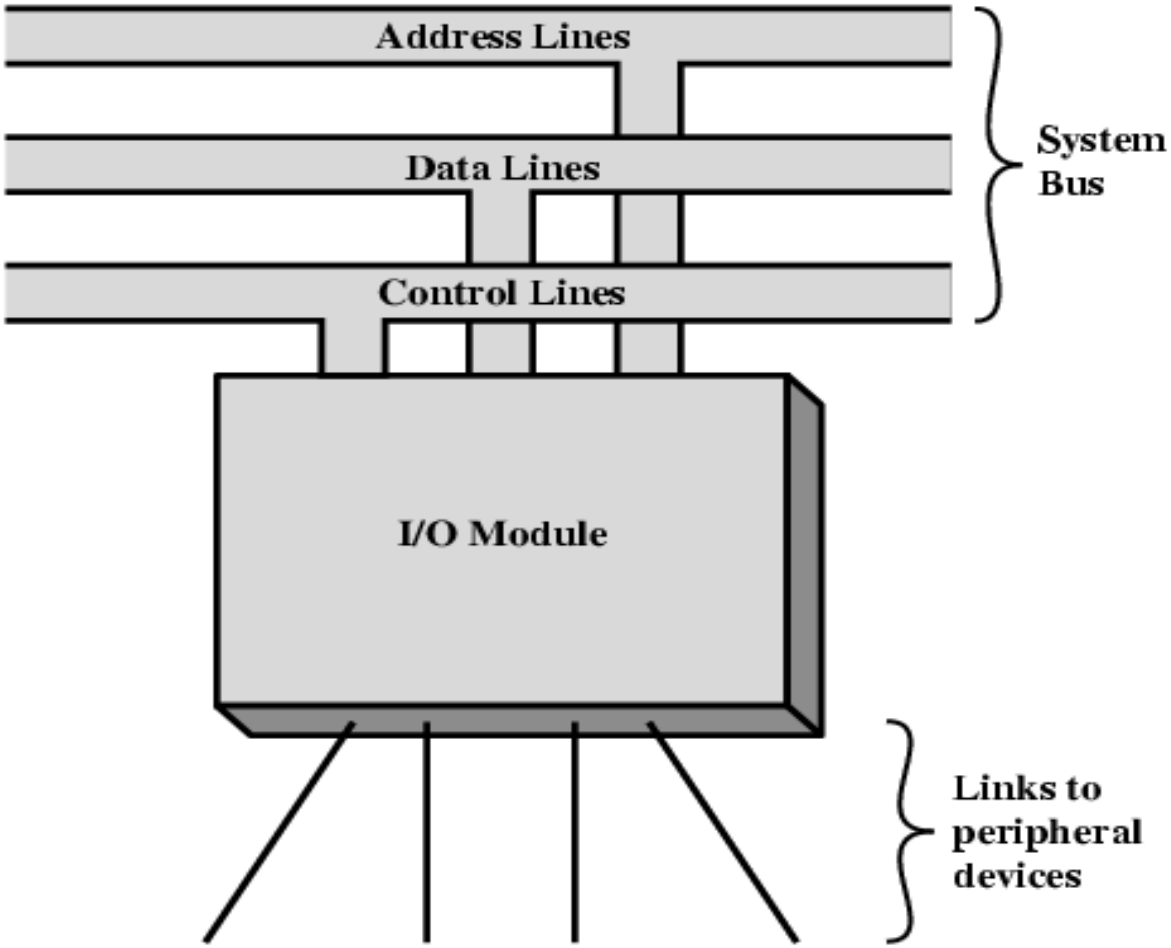
Input/Output Problems

A. Wide variety of peripherals
   - Delivering different amounts of data
   - At different speeds
   - In different formats
B. All slower than CPU and RAM
C. Thus, Need I/O modules

Input/Output Module

- This module has two major functions
  - Interface to the processor and memory via the system bus or central switch.
  - Interface to one or more peripheral devices by tailored data links.

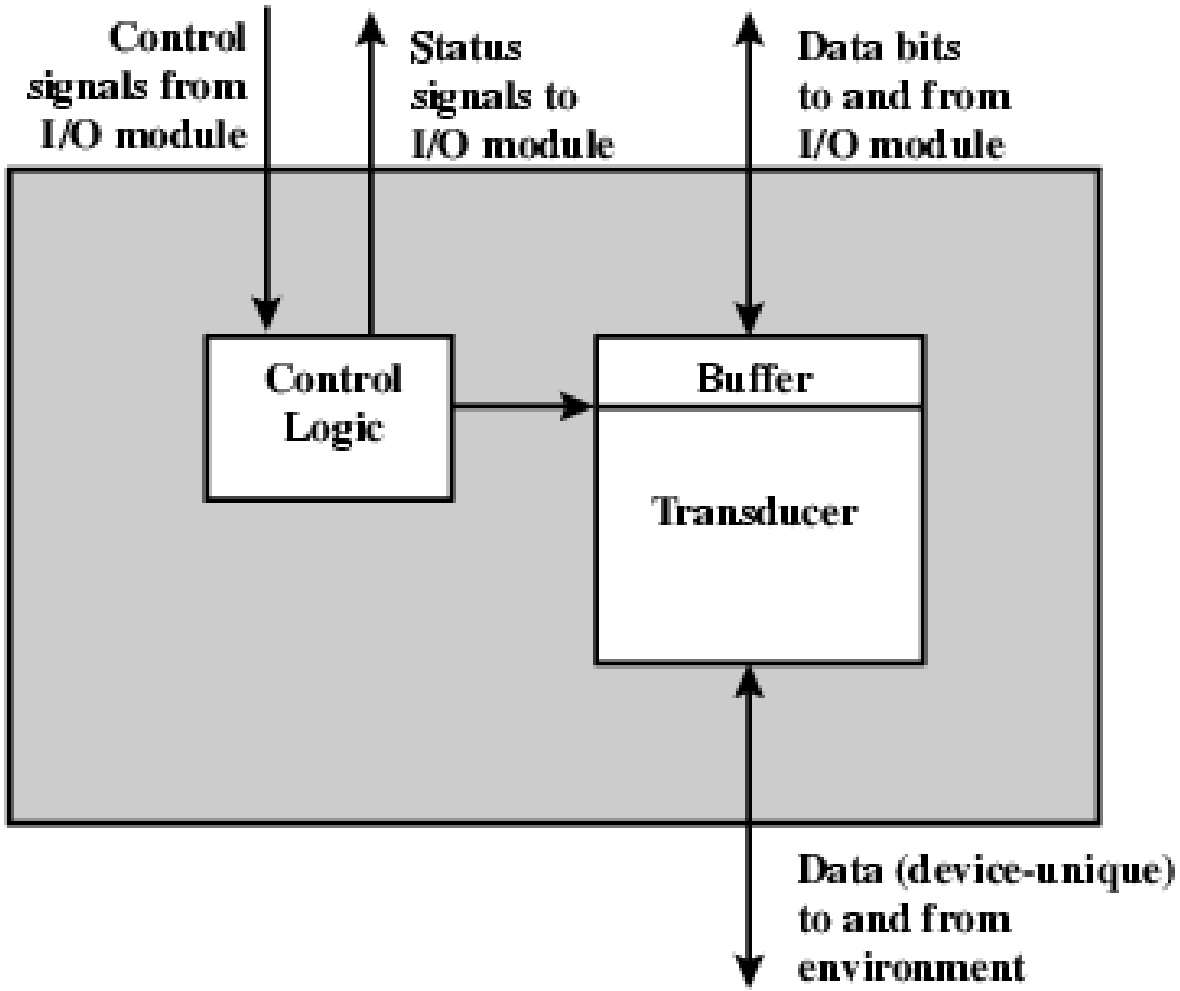
Generic Model of I/O Module



External Devices

- I/O operations are accomplished through a wide assortment of external devices that provide a means of exchanging data between the external environment and the computer.
- An external device attaches to the computer by a link to an I/O module.
- The link is used to exchange control, status, and data between the I/O module and the external device.
- An external device connected to an I/O module is often referred to as a *peripheral device* or, simply, a *peripheral*.
- Broadly classify external devices into three categories:
  - **Human readable:** Suitable for communicating with the computer user, e.g., Screen, printer, keyboard
  - **Machine readable:** Suitable for communicating with equipment, Monitoring and control
  - **Communication:** Suitable for communicating with remote devices, e.g., Modem, Network Interface Card (NIC).

External Device Block Diagram



I/O Module Function

- Control & Timing
- CPU Communication
- Device Communication
- Data Buffering
- Error Detection

I/O Steps

- CPU checks I/O module device status
- I/O module returns status
- If ready, CPU requests data transfer
- I/O module gets data from device
- I/O module transfers data to CPU
- Variations for output, DMA, etc.

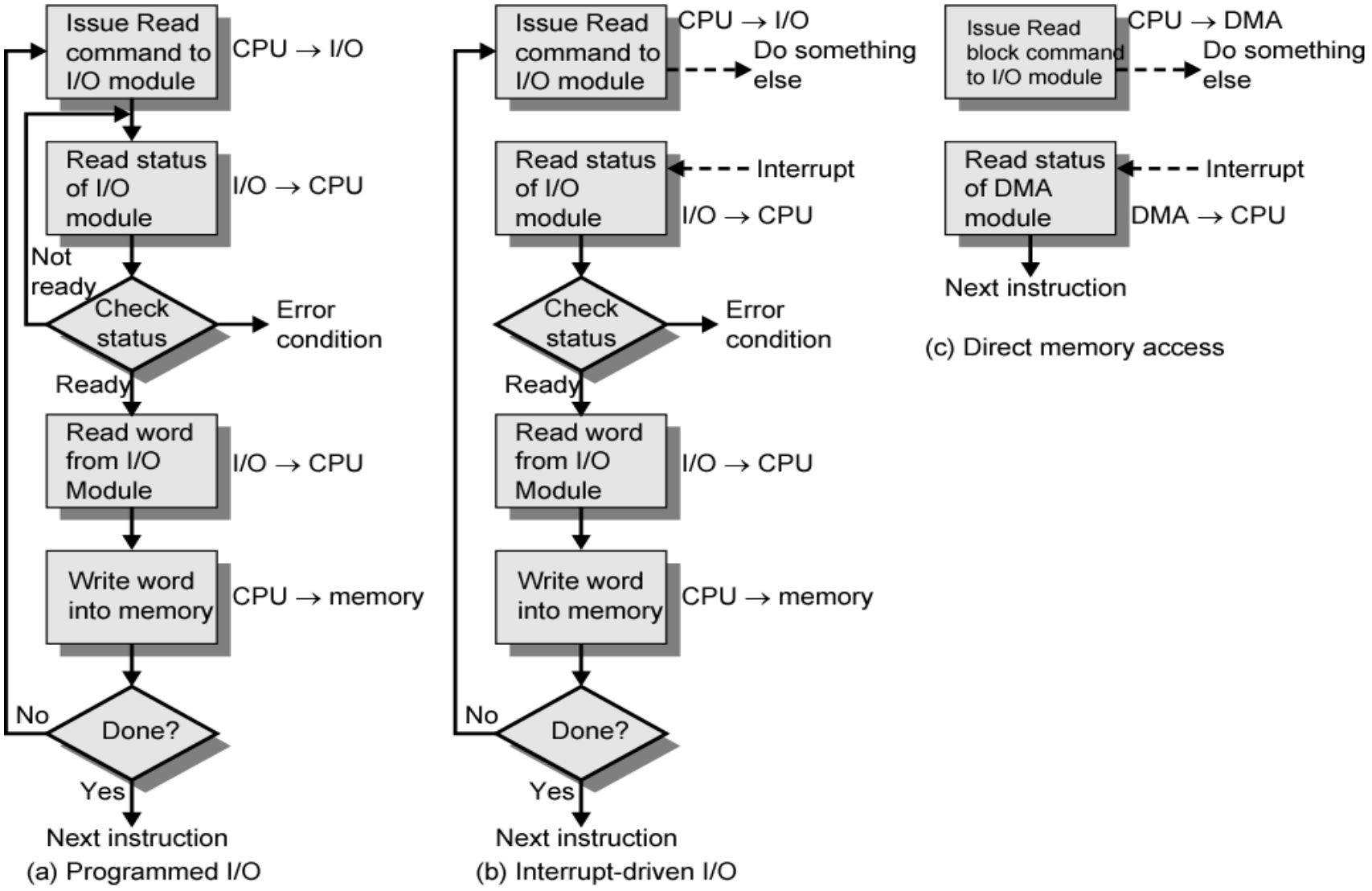
I/O Module Decisions

- Hide or reveal device properties to CPU
- Support multiple or single device
- Control device functions or leave for CPU
- O/S decisions, e.g., Unix treats everything it can as a file

Input /Output Techniques

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

Three Techniques for Input of a Block of Data



Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

Programmed I/O - Overview

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

I/O Commands

- To execute an I/O-related instruction, the processor issues an address, specifying the particular I/O module and external device, and an I/O command.
- There are four types of I/O commands that an I/O module may receive when it is addressed by a processor: Control, Test, Read, and Write.
- Control - telling module what to do, e.g., spin up disk
- Test - check status, e.g., power? Error?
- Read/Write - Module transfers data via buffer from/to device

Addressing I/O Devices

- Under programmed I/O data transfer is very like memory access (from CPU viewpoint)
- Each device is given unique identifier
- CPU commands contain identifier (address)

I/O Mapping

- With programmed I/O, there is a close correspondence between the I/O-related instructions that the processor fetches from memory and the I/O commands that the processor issues to an I/O module to execute the instructions.
- When the processor, main memory, and I/O share a common bus, two modes of addressing are possible: memory mapped and isolated.
- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O - Limited set

Interrupt Driven I/O

- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

Interrupt Driven I/O - Basic Operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Design Issues

- How do you identify the module issuing the interrupt?
- How do you deal with multiple interrupts?
  - i.e. an interrupt handler being interrupted
- Four general categories of techniques are in common use:
  a) Multiple interrupt lines
  b) Software poll
  c) Daisy chain (hardware poll, vectored)
  d) Bus arbitration (vectored)

Identifying Interrupting Module

A. Multiple interrupt lines - Different line for each module
   - Each interrupt line has a priority
   - Higher priority lines can interrupt lower priority lines
   - If bus mastering only current master can interrupt
   - PC
   - Limits number of devices

B. Software poll
   - CPU asks each module in turn
   - Slow

Identifying Interrupting Module Contd…

C. Daisy Chain or Hardware poll
- Interrupt Acknowledge sent down a chain
- Module responsible places vector on bus
- CPU uses vector to identify handler routine

D. Bus Master
- Module must claim the bus before it can raise interrupt
- e.g. PCI & SCSI

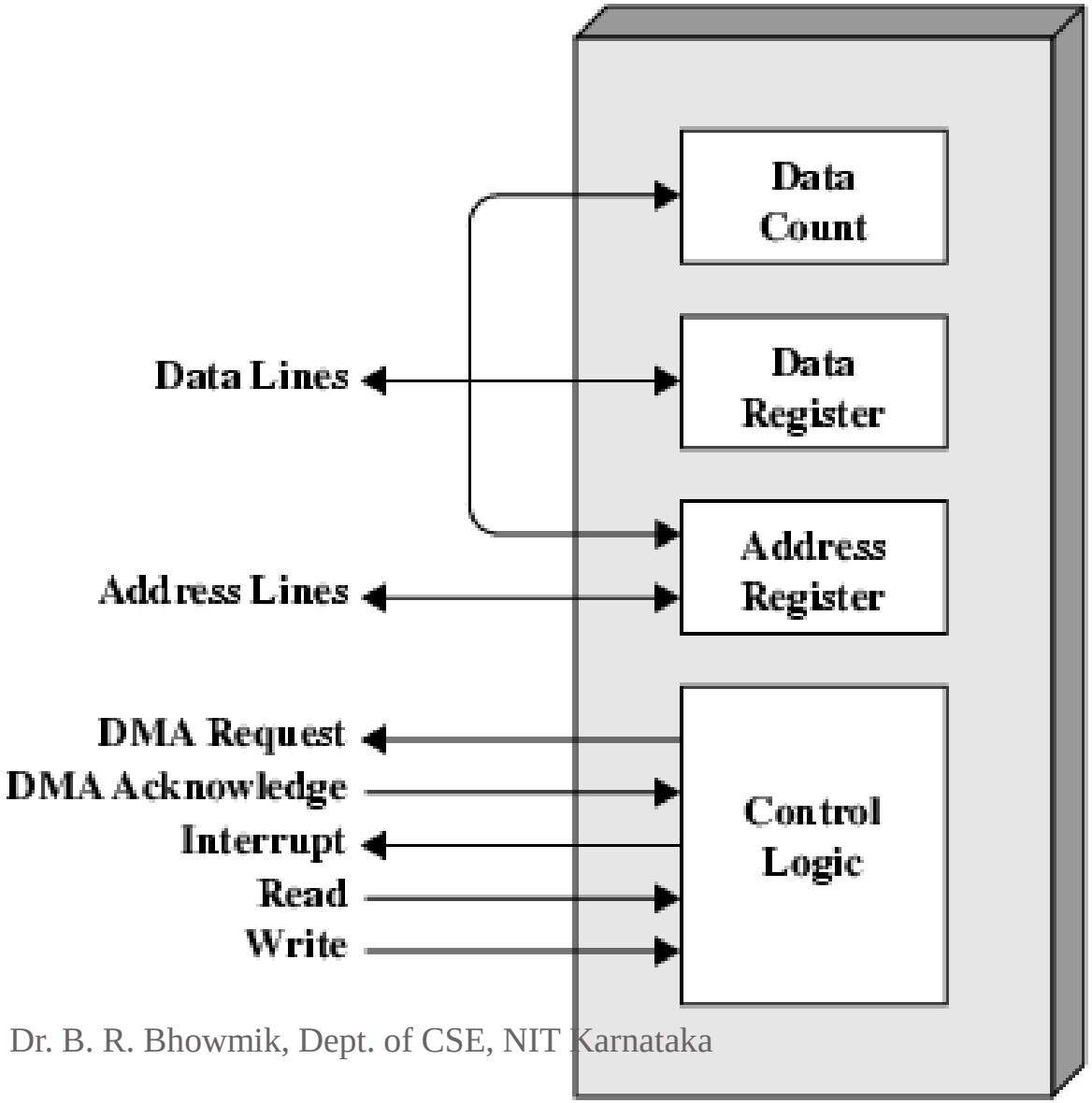
Direct Memory Access

- Programmed and Interrupt-Driven I/O suffer from two inherent drawbacks:
  a) Limited Transfer rate - The I/O transfer rate is limited by the speed with which the processor can test and service a device.
  b) Tied-up CPU - The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.
- Interrupt driven and programmed I/O require active CPU intervention .
- DMA is the answer

DMA Function

- Additional Module (hardware) on bus
- DMA controller takes over from CPU for I/O

Typical DMA Module Diagram



DMA Operation

- CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for data
  - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

DMA Transfer Cycle Stealing

- DMA involves an additional module on the system bus.
- The DMA module is capable of mimicking the processor and, indeed, of taking over control of the system from the processor.
- It needs to do this to transfer data to and from memory over the system bus.
- For this purpose, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.
- The latter technique is more common and is referred to as *cycle stealing,* because the DMA module in effect steals a bus cycle.
- DMA controller in effect steals a bus cycle.
- Not an interrupt - CPU does not switch context.
- CPU suspended just before it accesses bus, i.e., before an operand or data fetch or a data write.
- Slows down CPU but not as much as CPU doing transfer.

Thank You