Chapter 5
Input/Output

External devices

- Provide a means of exchanging data between the external environment and the computer
- Attach to the computer by a link to an I/O module
- Three categories:
  - Human readable: Suitable for communicating with the computer user (e.g. video display terminals, printers)
  - Machine readable: Suitable for communicating with equipment (e.g. magnetic disk and tape systems, sensors and actuator)
  - Communication: Suitable for communicating with remote devices such as a terminal, a machine readable device, or another computer

Keyboard/Monitor

- User provides input through the keyboard
- The monitor displays data provided by the computer
- The basic unit of exchange is the character; each associated with a code, typically 7 or 8 bits in length.
- The most commonly used text code is the International Reference Alphabet (IRA) in 7-bit binary code
- Characters are of two types: printable and control

Why to use I/O modules?

- There are a wide variety of peripherals with various methods of operation. It would be impractical to incorporate the necessary logic within the processor to control a range of devices.
- The data transfer rate of peripherals is often much slower than that of the memory or processor. Thus, it is impractical to use the high-speed system bus to communicate directly with a peripheral.
- The data transfer rate of some peripherals can be faster than that of the memory or processor. Again, the mismatch would lead to inefficiencies if not managed properly.
- Peripherals often use different data formats and word lengths than the computer to which they are attached.

I/O modules

- What an I/O module is used for?
  - Interface to the processor and memory via the system bus or central switch
  - Interface to one or more peripheral devices by tailored data links

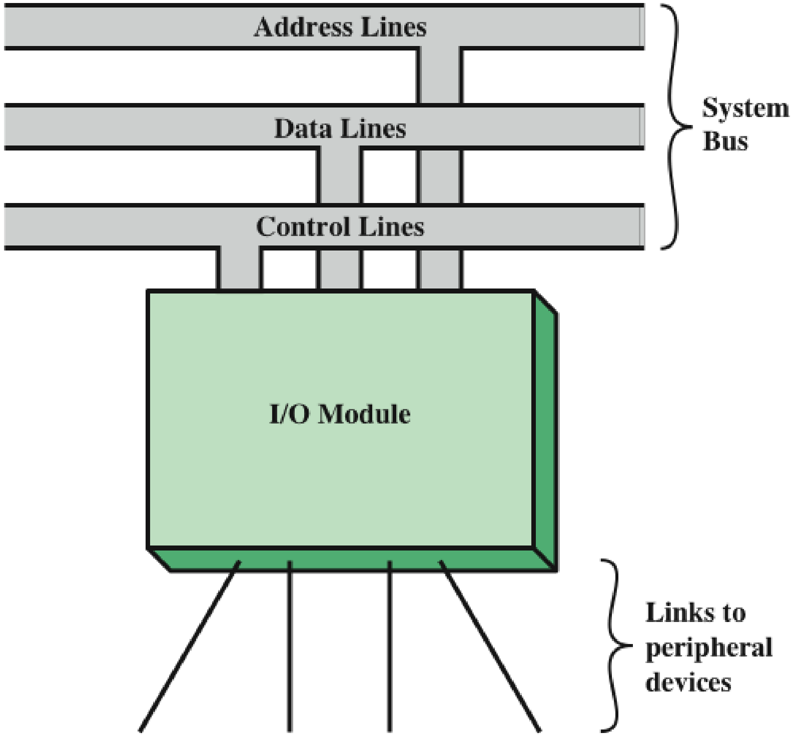

Figure 7.1 Generic Model of an I/O Module

I/O module functions

The major functions for an I/O module fall into the following categories:

Control and timing

- Coordinates the flow of traffic between internal resources and external devices

Processor communication

- Involves command decoding, data, status reporting, address recognition

Device communication

- Involves commands, status information, and data

Data buffering

- Performs the needed buffering operation to balance device and memory speeds

Error detection

- Detects and reports transmission errors

I/O module structure

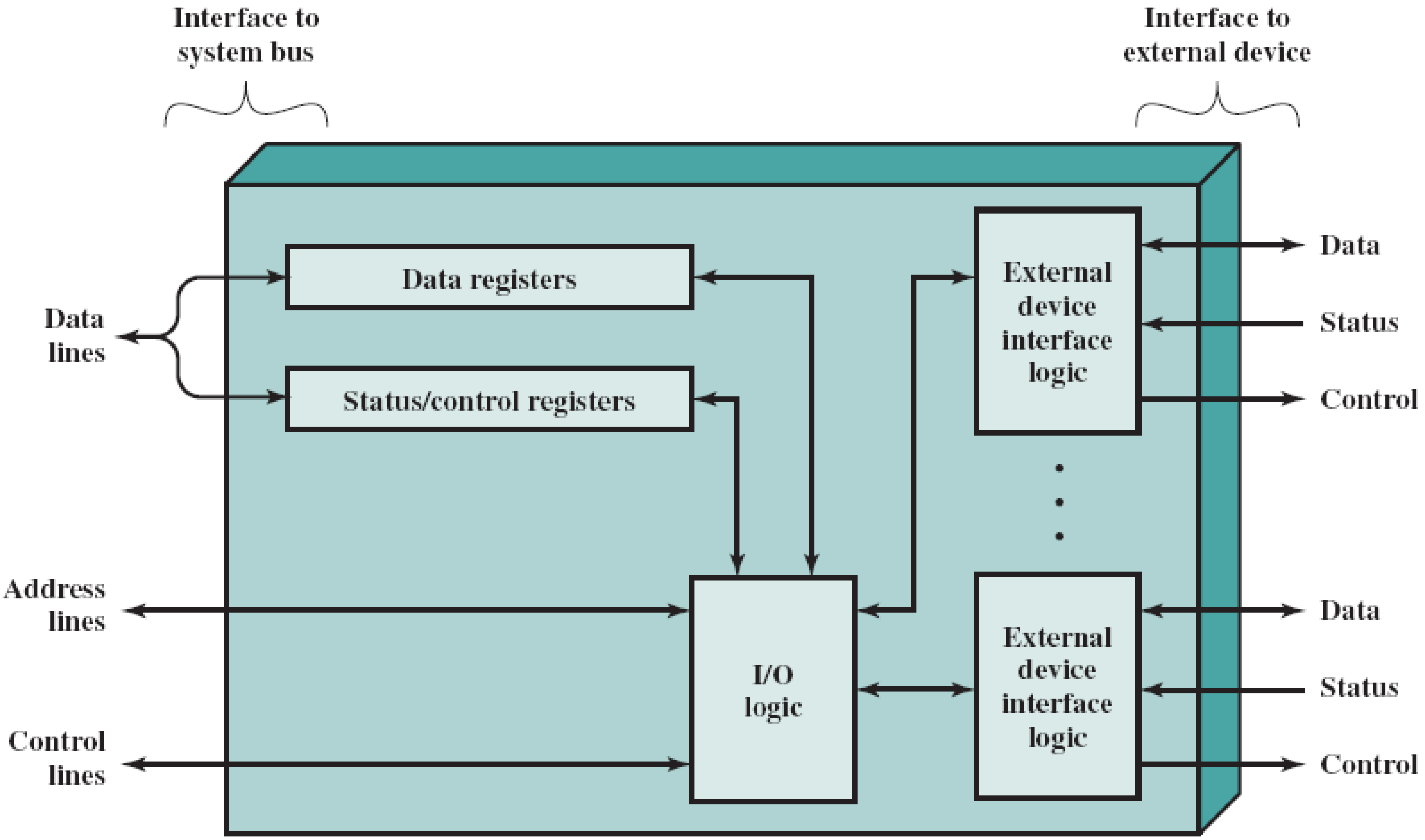

Figure 7.3 Block Diagram of an I/O Module

I/O commands

- There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:
  - Control: used to activate a peripheral and tell it what to do
  - Test: used to test various status conditions associated with an I/O module and its peripherals
  - Read: causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer
  - Write: causes the I/O module to take an item of data from the data bus and subsequently transmit that data item to the peripheral

I/O mapping

- 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
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O

I/O Operation Techniques

- Programmed I/O
  - Data are exchanged between the processor and the I/O module
  - Processor executes a program that gives it direct control of the I/O operation
  - When the processor issues a command it must wait until the I/O operation is complete
  - If the processor is faster than the I/O module this is wasteful of processor time
- Interrupt-driven I/O
  - Processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work
- Direct Memory Access
  - The I/O module and main memory exchange data directly without processor involvement

Drawbacks of Programmed and Interrupt-Driven I/O

- The I/O transfer rate is limited by the speed with which the processor can test and service a device
- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer

→ When large volumes of data are to be moved a more efficient technique is direct memory access (DMA)

DMA Module

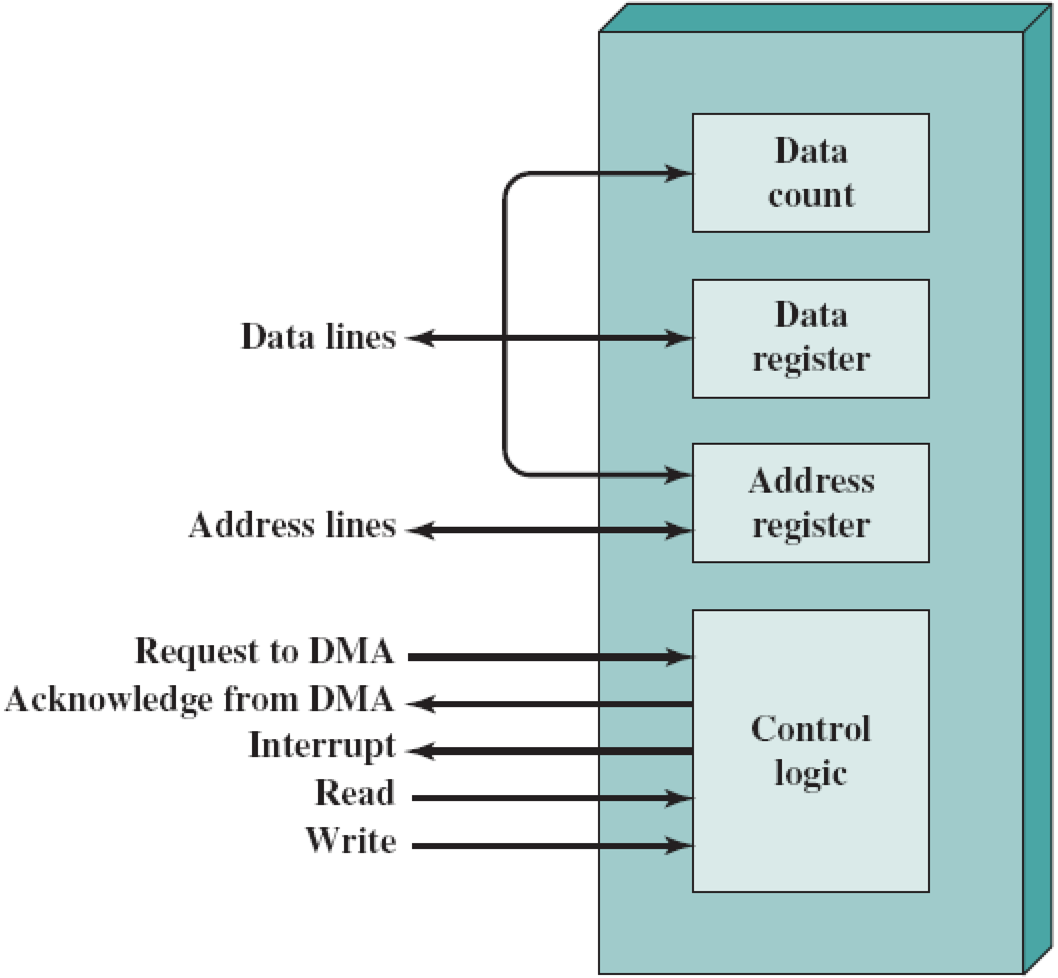

Figure 7.11 Typical DMA Block Diagram

Device Identification

- Multiple interrupt lines
  - Between the processor and the I/O modules
  - Most straightforward approach to the problem
  - Consequently even if multiple lines are used, it is likely that each line will have multiple I/O modules attached to it
- Software poll
  - When processor detects an interrupt it branches to an interrupt-service routine whose job is to poll each I/O module to determine which module caused the interrupt
  - Time consuming

Device Identification (Cont'd)

- Daisy chain (hardware poll, vectored)
  - The interrupt acknowledge line is daisy chained through the modules
  - Vector – address of the I/O module or some other unique identifier
  - Vectored interrupt – processor uses the vector as a pointer to the appropriate device-service routine, avoiding the need to execute a general interrupt-service routine first
- Bus arbitration (vectored)
  - An I/O module must first gain control of the bus before it can raise the interrupt request line
  - When the processor detects the interrupt it responds on the interrupt acknowledge line
  - Then the requesting module places its vector on the data lines

Evolution of I/O function

1. The CPU directly controls a peripheral device. This is seen in simple microprocessor-controlled devices.
2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts.
3. The same configuration as in step 2 is used, but now interrupts are employed. The CPU need not spend time waiting for an I/O operation to be performed, thus increasing efficiency.
4. The I/O module is given direct access to memory via DMA.
5. The I/O module is enhanced to become a processor in its own right, with a 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.