

# Computer Architecture & Organization

## Chapter 14

### Input/Output

Generic Model of an I/O Module

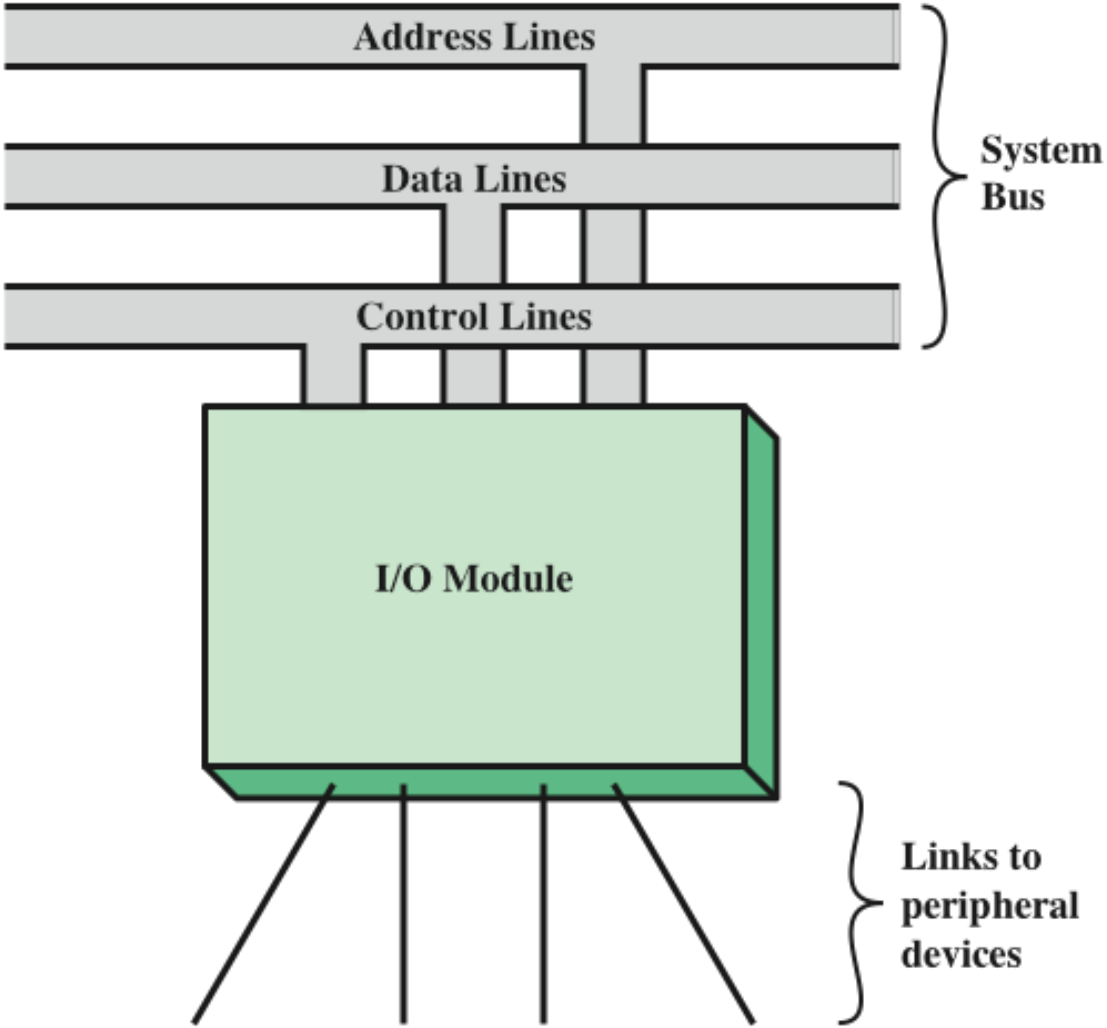


Figure 7.1 Generic Model of an I/O Module

# 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
  - The link is used to exchange control, status, and data between the I/O module and the external device
- *peripheral device*
  - An external device connected to an I/O module
- Three categories:
- Human readable
  - Suitable for communicating with the computer user
  - Video display terminals (VDTs), printers
- Machine readable
  - Suitable for communicating with equipment
  - Magnetic disk and tape systems, sensors and actuators
- Communication
  - Suitable for communicating with remote devices such as a terminal, a machine readable device, or another computer

# External Device Block Diagram

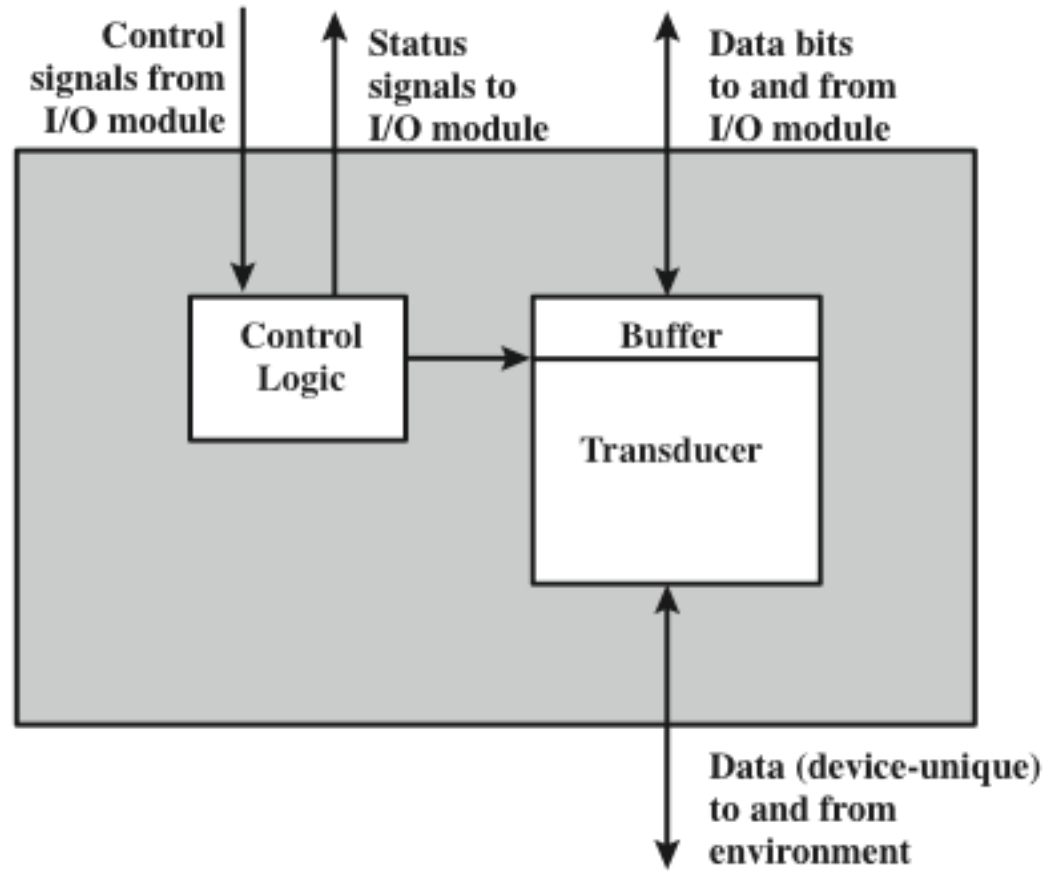


Figure 7.2 Block Diagram of an External Device

# Keyboard/Monitor

## International Reference Alphabet (IRA)

- Basic unit of exchange is the character
  - Associated with each character is a code
  - Each character in this code is represented by a unique 7-bit binary code
    - 128 different characters can be represented
- Characters are of two types:
  - Printable
    - Alphabetic, numeric, and special characters that can be printed on paper or displayed on a screen
  - Control
    - Have to do with controlling the printing or displaying of characters
    - Example is carriage return
    - Other control characters are concerned with communications procedures

Most common means of computer/user interaction

User provides input through the keyboard

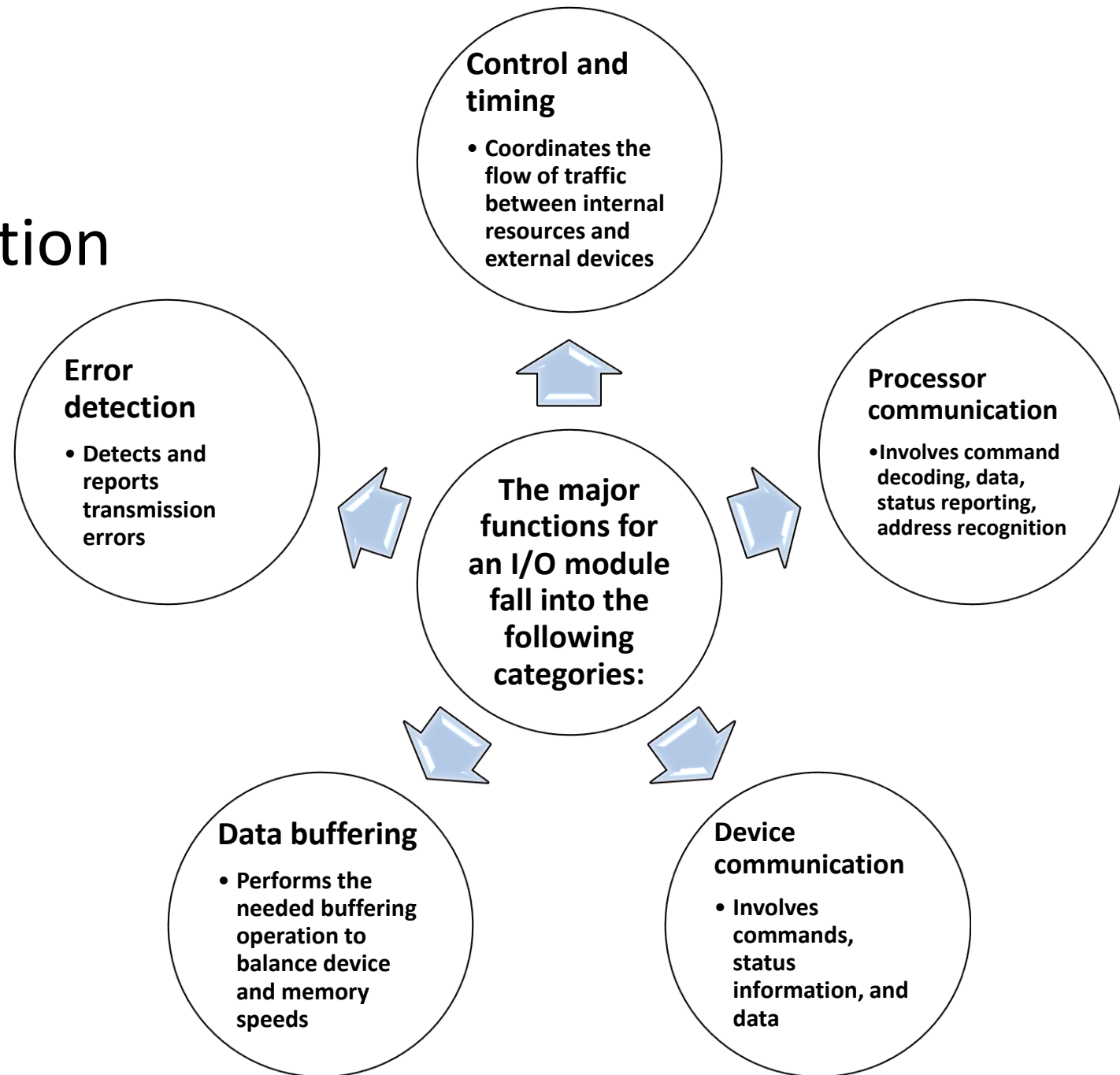
The monitor displays data provided by the computer

## Keyboard Codes

- When the user depresses a key it generates an electronic signal that is interpreted by the transducer in the keyboard and translated into the bit pattern of the corresponding IRA code
- This bit pattern is transmitted to the I/O module in the computer
- On output, IRA code characters are transmitted to an external device from the I/O module
- The transducer interprets the code and sends the required electronic signals to the output device either to display the indicated character or perform the requested control function

# I/O Modules

## Module Function



# I/O Module Structure

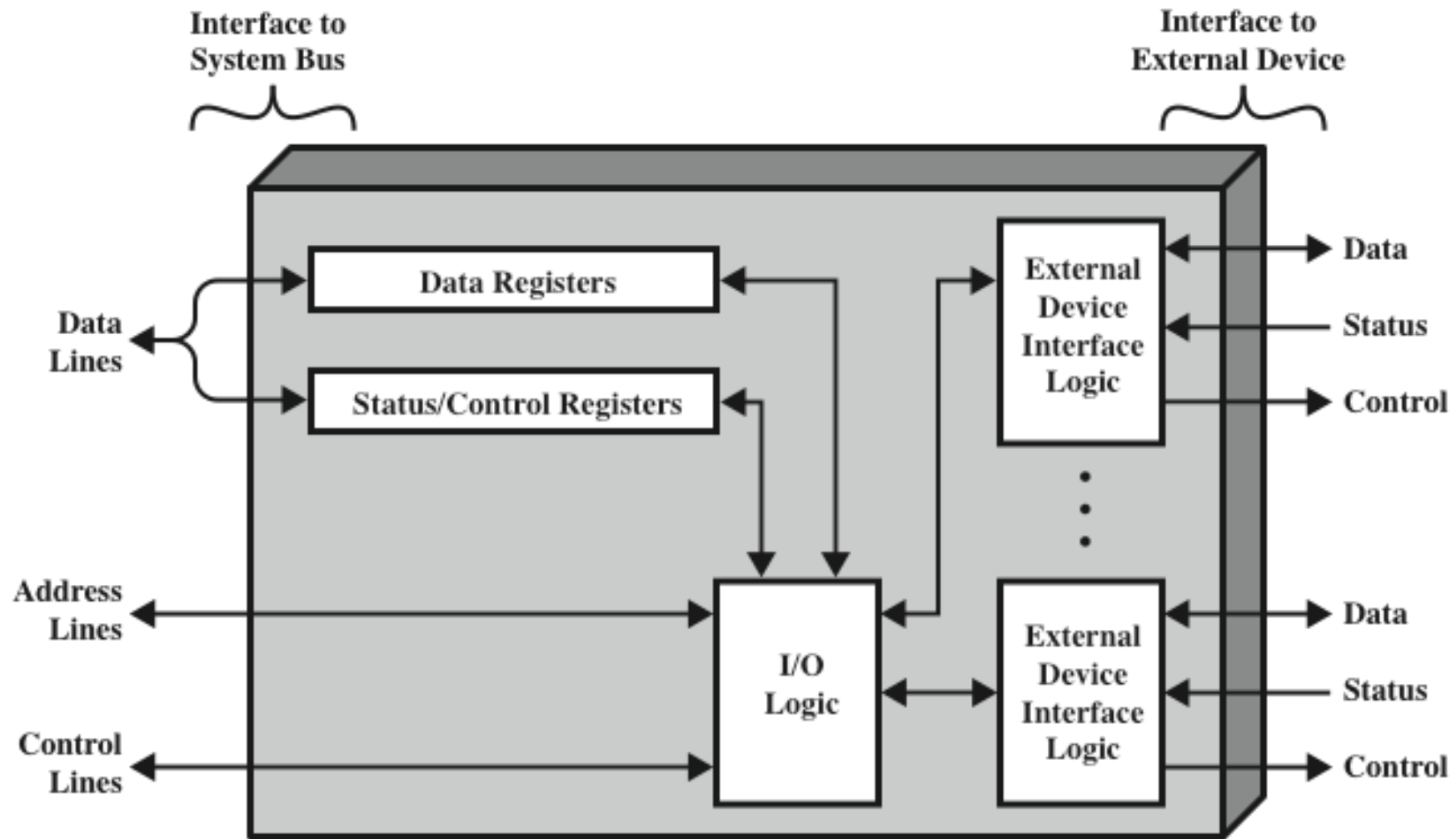


Figure 7.3 Block Diagram of an I/O Module

# Programmed I/O

- Three techniques are possible for I/O operations:
- 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 (DMA)
  - The I/O module and main memory exchange data directly without processor involvement



# 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:
  - 1) Control
    - used to activate a peripheral and tell it what to do
  - 2) Test
    - used to test various status conditions associated with an I/O module and its peripherals
  - 3) Read
    - causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer
  - 4) 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 Instructions

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

The form of the instruction depends on the way in which external devices are addressed

Each I/O device connected through I/O modules is given a unique identifier or address

When the processor issues an I/O command, the command contains the address of the desired device

Thus each I/O module must interpret the address lines to determine if the command is for itself

## Memory-mapped I/O

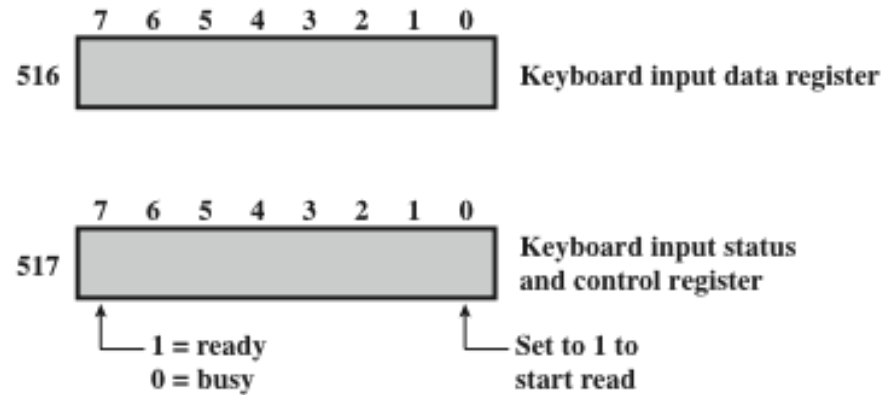
There is a single address space for memory locations and I/O devices

A single read line and a single write line are needed on the bus

# I/O Mapping Summary

- 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

- Figure 7.5a shows how the interface for a simple input device such as a terminal keyboard might appear to a programmer using memory-mapped I/O.
- Assume a 10-bit address, with a 512-bit memory (locations 0–511) and up to 512 I/O addresses (locations 512–1023).
- Two addresses are dedicated to keyboard input from a particular terminal. Address 516 refers to the data register and address 517 refers to the status register, which also functions as a control register for receiving processor commands.
- With isolated I/O (Figure 7.5b), the I/O ports are accessible only by special I/O commands, which activate the I/O command lines on the bus.
- Advantage of memory-mapped I/O is that those large repertoire of instructions can be used, allowing more efficient programming. A disadvantage is that valuable memory address space is used up.



Memory  
Mapped  
I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte


(b) Isolated I/O

Isolated  
I/O


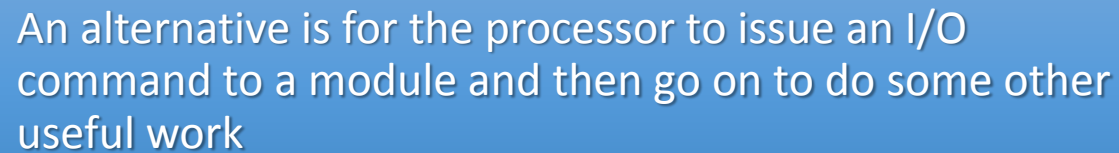
Figure 7.5 Memory-Mapped and Isolated I/O

# Interrupt-Driven I/O


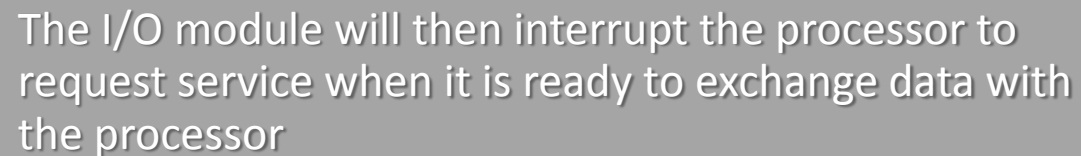
The problem with programmed I/O is that the processor has to wait a long time for the I/O module to be ready for either reception or transmission of data



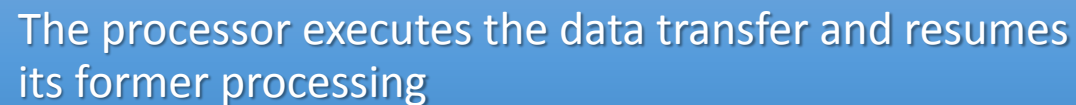
An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work



The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor



The processor executes the data transfer and resumes its former processing



# Changes in Memory and Registers for an Interrupt

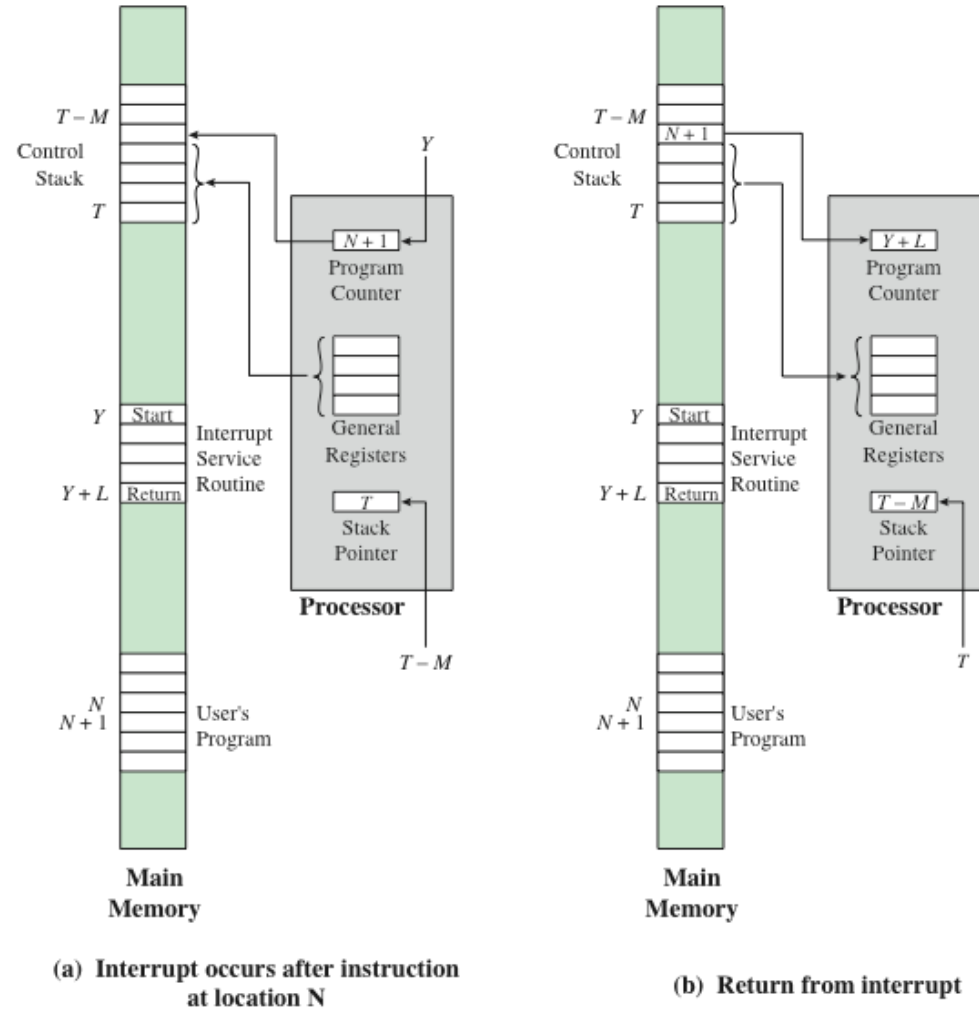



Figure 7.7 Changes in Memory and Registers for an Interrupt

# Design Issues



Two design issues arise in implementing interrupt I/O:

- Because there will be multiple I/O modules how does the processor determine which device issued the interrupt?
- If multiple interrupts have occurred how does the processor decide which one to process?

# + Device Identification

Four general categories of techniques are in common use:

- **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 borrowed**
  - 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
- **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



# Intel 82C59A Interrupt Controller

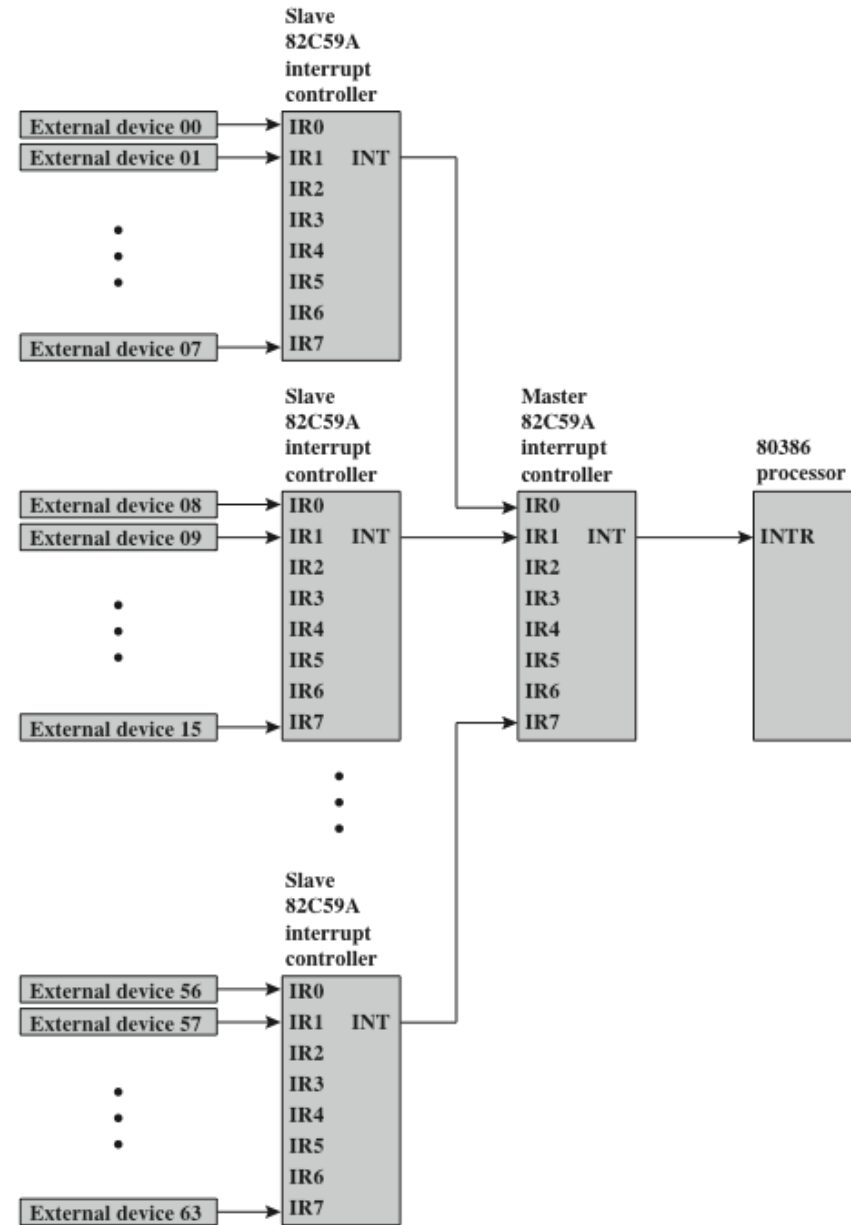


Figure 7.8 Use of the 82C59A Interrupt Controller

# Intel 82C55A

## Programmable Peripheral Interface

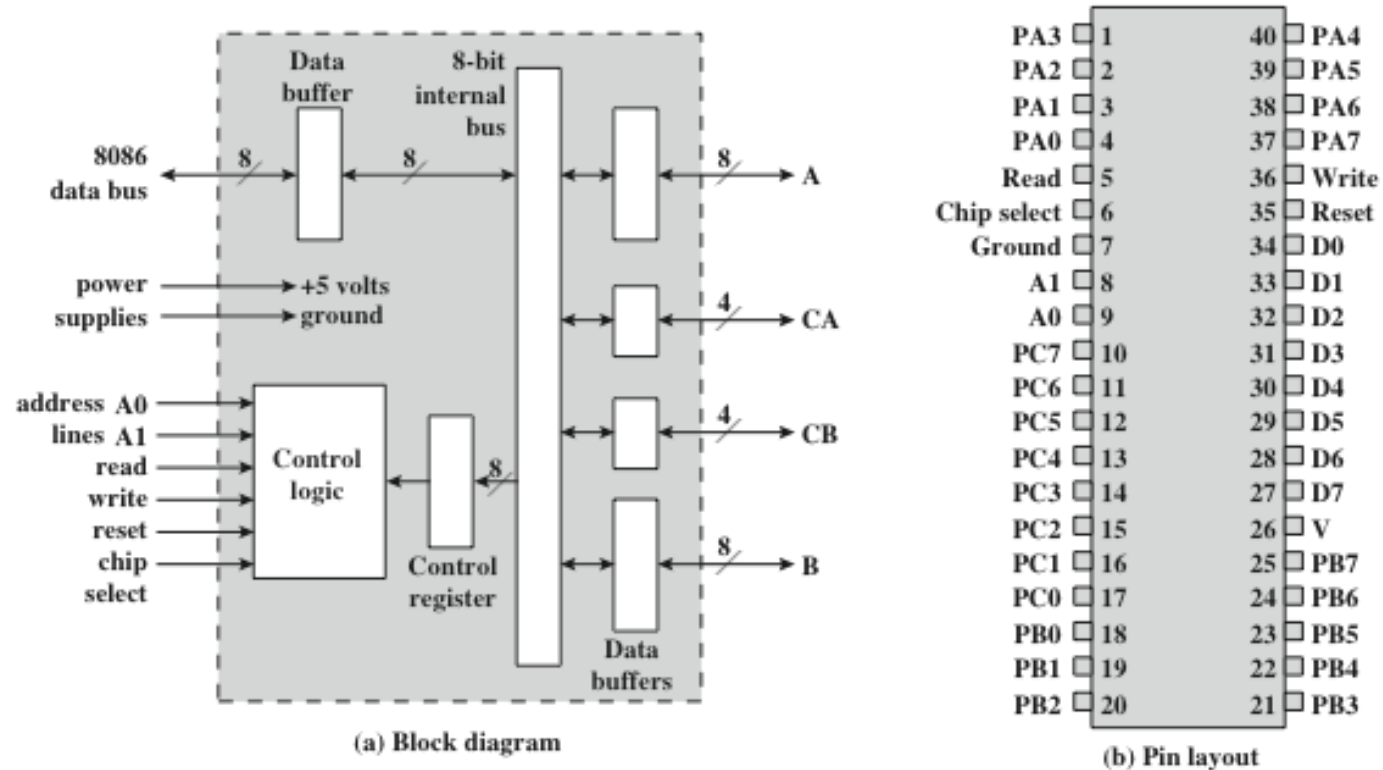


Figure 7.9 The Intel 82C55A Programmable Peripheral Interface

# Keyboard/Display Interfaces to 2C55A

- Because the 82C55A is programmable via the control register, it can be used to control a variety of simple peripheral devices.
- The keyboard provides 8 bits of input. Two of these bits, SHIFT and CONTROL, have special meaning to the keyboard-handling program executing in the processor. However, this interpretation is transparent to the 82C55A,
- Two handshaking control lines are provided for use with the keyboard.
- The display is also linked by an 8-bit data port. Again, two of the bits have special meanings that are transparent to the 82C55A.
- In addition to two handshaking lines, two lines provide additional control functions.

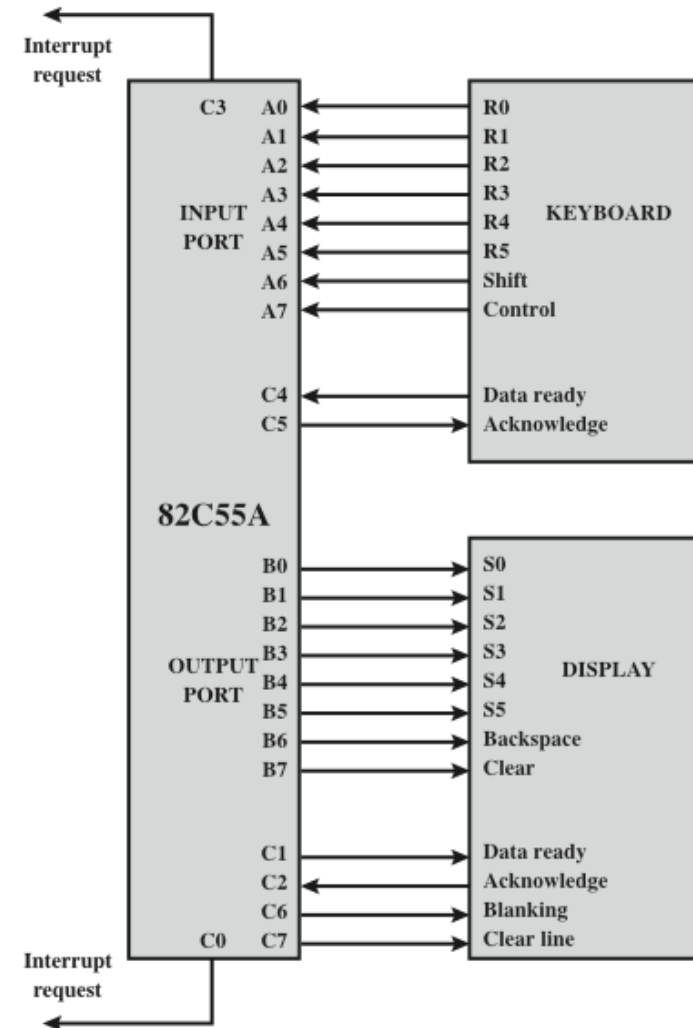


Figure 7.10 Keyboard/Display Interface to 82C55A

# Drawbacks of Programmed and Interrupt-Driven I/O

- Both forms of I/O suffer from two inherent drawbacks:
  - 1) The I/O transfer rate is limited by the speed with which the processor can test and service a device
  - 2) 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)

# Typical DMA Module Diagram

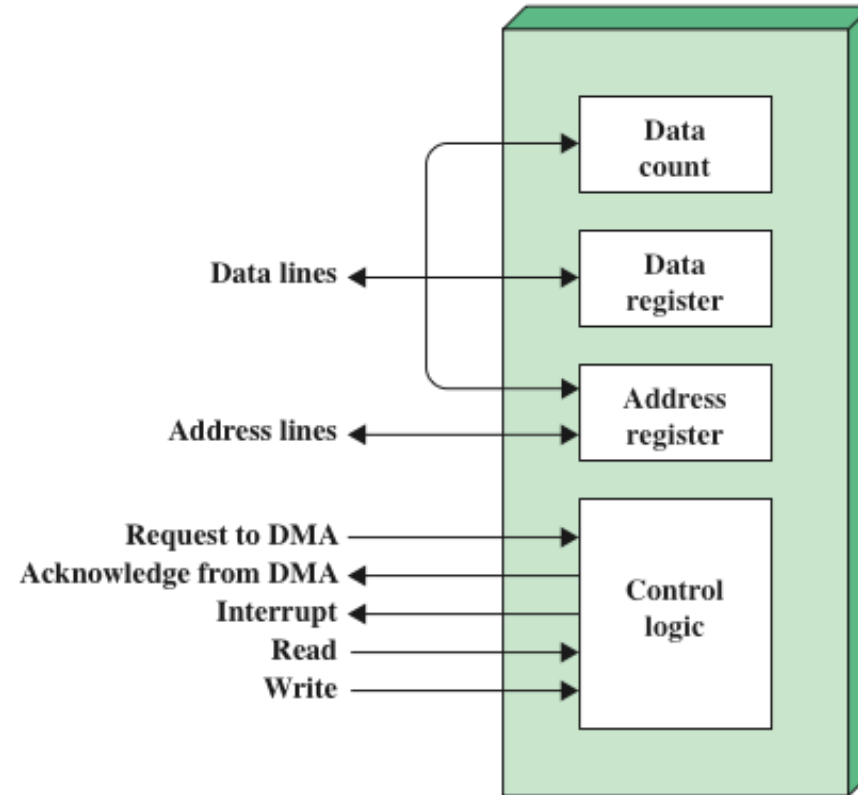
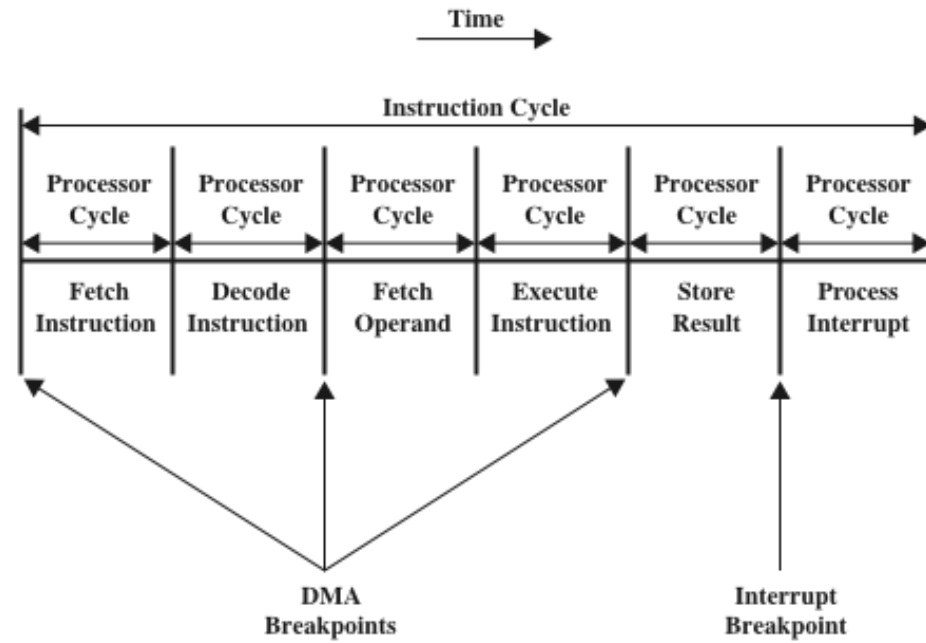


Figure 7.11 Typical DMA Block Diagram



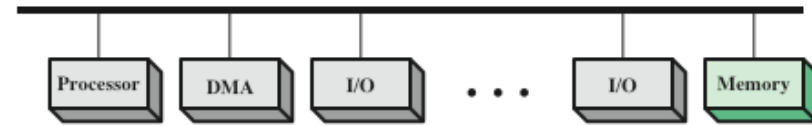
DMA

DMA

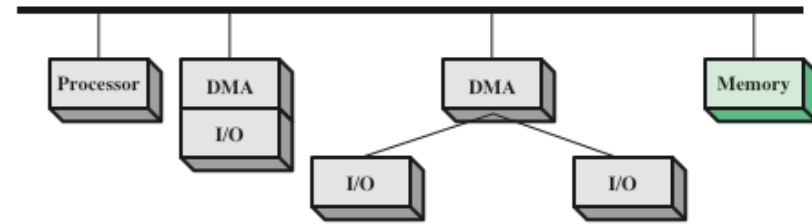
Figure 7.12 DMA and Interrupt Breakpoints During an Instruction Cycle

DMA Operation

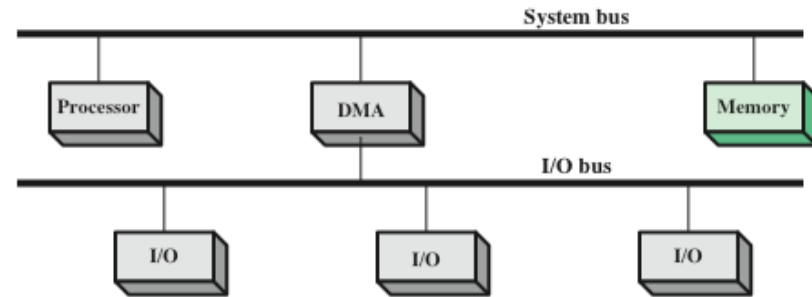
# Alternative DMA Configurations



(a) Single-bus, detached DMA



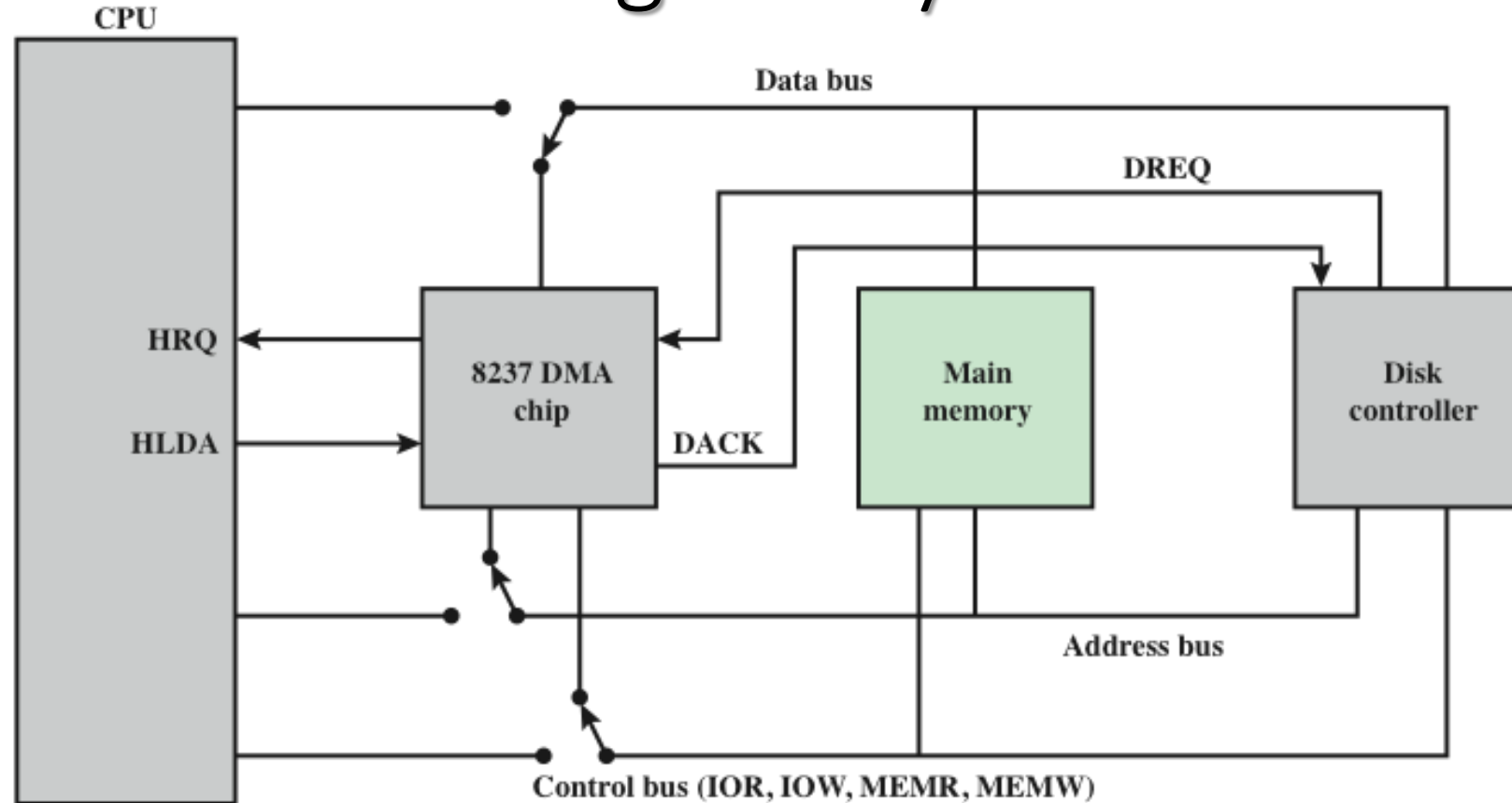
(b) Single-bus, Integrated DMA-I/O



(c) I/O bus

Figure 7.13 Alternative DMA Configurations

# 8237 DMA Usage of System Bus



DACK = DMA acknowledge  
DREQ = DMA request  
HLDA = HOLD acknowledge  
HRQ = HOLD request

Figure 7.14 8237 DMA Usage of System Bus



# Fly-By DMA Controller

Data does not pass through and is not stored in DMA chip

- DMA only between I/O port and memory
- Not between two I/O ports or two memory locations

Can do memory to memory via register

8237 contains four DMA channels

- Programmed independently
- Any one active
- Numbered 0, 1, 2, and 3

Bit	Command	Status	Mode	Single Mask	All Mask
D0	Memory-to-memory E/D	Channel 0 has reached TC	Channel select	Select channel mask bit	Clear/set channel 0 mask bit
D1	Channel 0 address hold E/D	Channel 1 has reached TC			Clear/set channel 1 mask bit
D2	Controller E/D	Channel 2 has reached TC	Verify/write/read transfer	Clear/set mask bit	Clear/set channel 2 mask bit
D3	Normal/compressed timing	Channel 3 has reached TC			Clear/set channel 3 mask bit
D4	Fixed/rotating priority	Channel 0 request	Auto-initialization E/D	Not used	Not used
D5	Late/extended write selection	Channel 0 request	Address increment/decrement select		
D6	DREQ sense active high/low	Channel 0 request			
D7	DACK sense active high/low	Channel 0 request	Demand/single/block/cascade mode select		

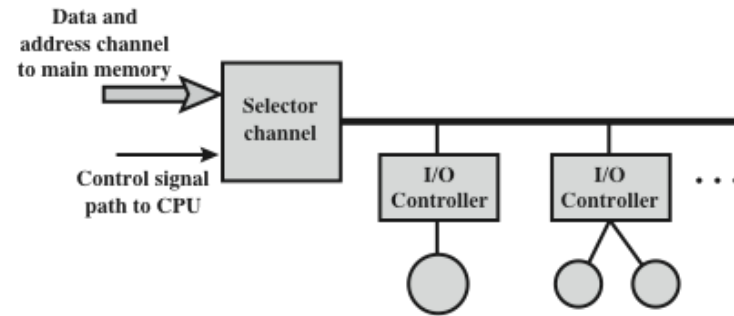
E/D = enable/disable  
TC = terminal count

Table 7.2  
Intel  
8237A Registers

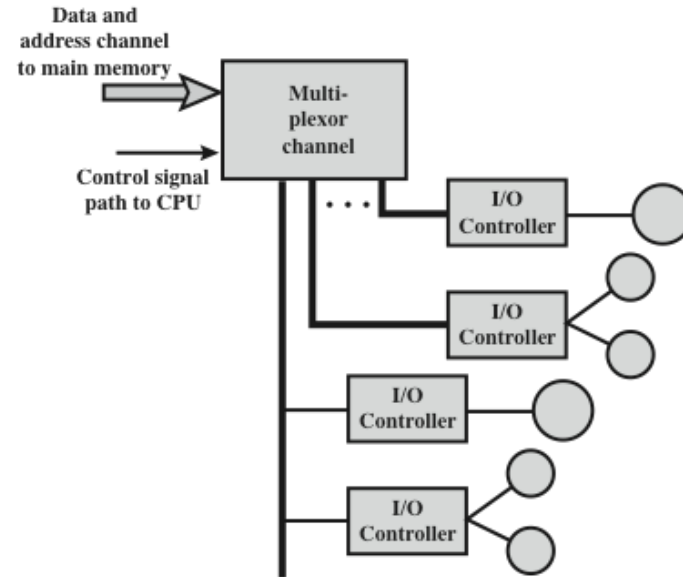
# Evolution of the I/O Function

1. The CPU directly controls a peripheral device.
2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts.
3. 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. It can now move a block of data to or from memory without involving the CPU, except at the beginning and end of the transfer.
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. With this architecture a large set of I/O devices can be controlled with minimal CPU involvement.

# I/O Channel Architecture



(a) Selector



(b) Multiplexor

Figure 7.15 I/O Channel Architecture

# Parallel and Serial I/O

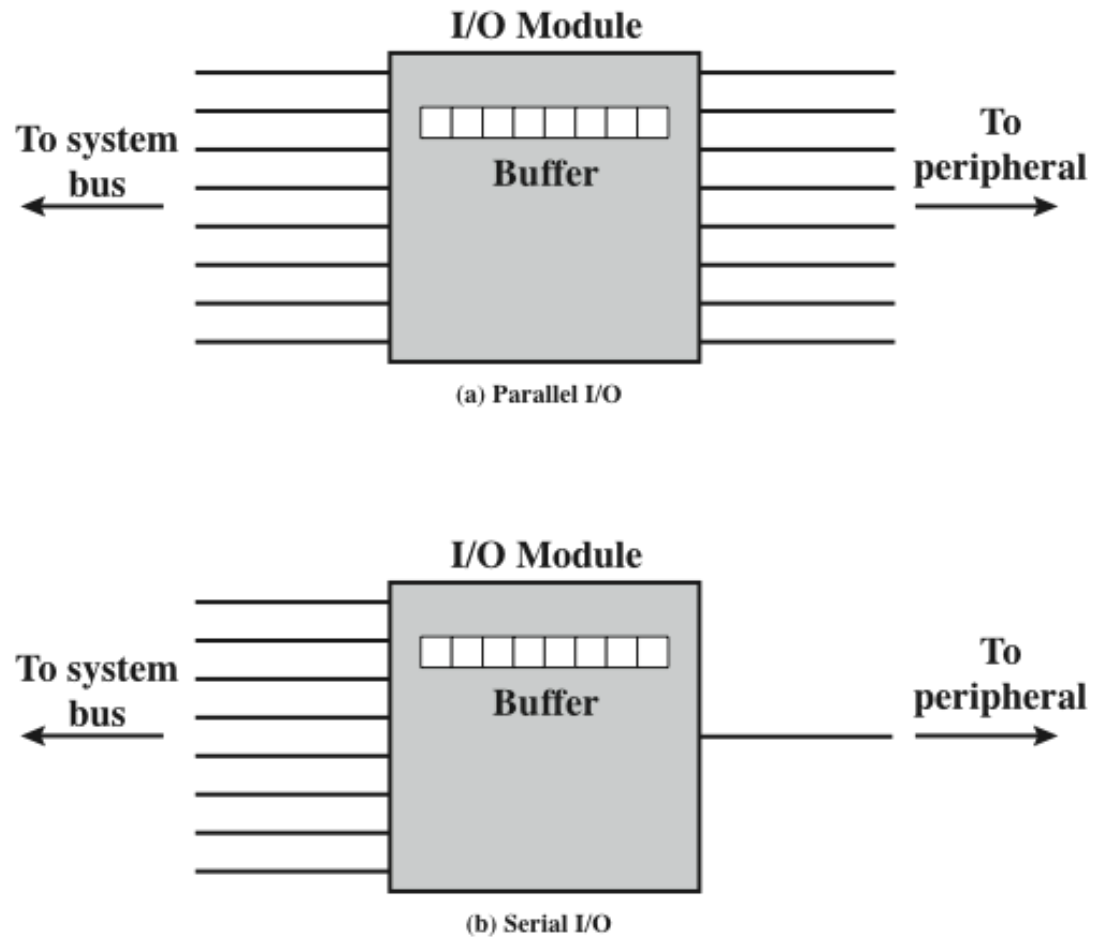
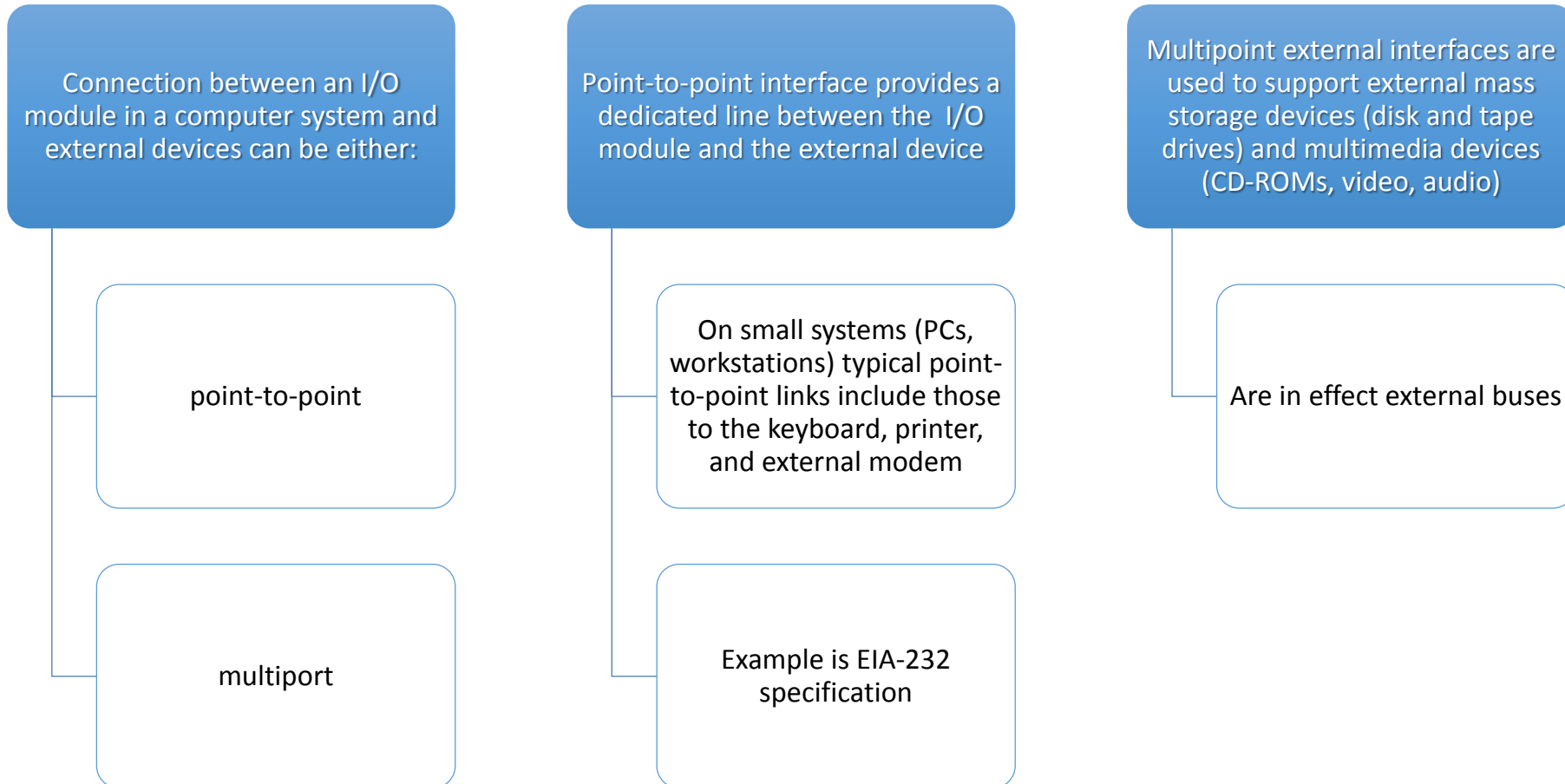


Figure 7.16 Parallel and Serial I/O

# Point-to-Point and Multipoint Configurations

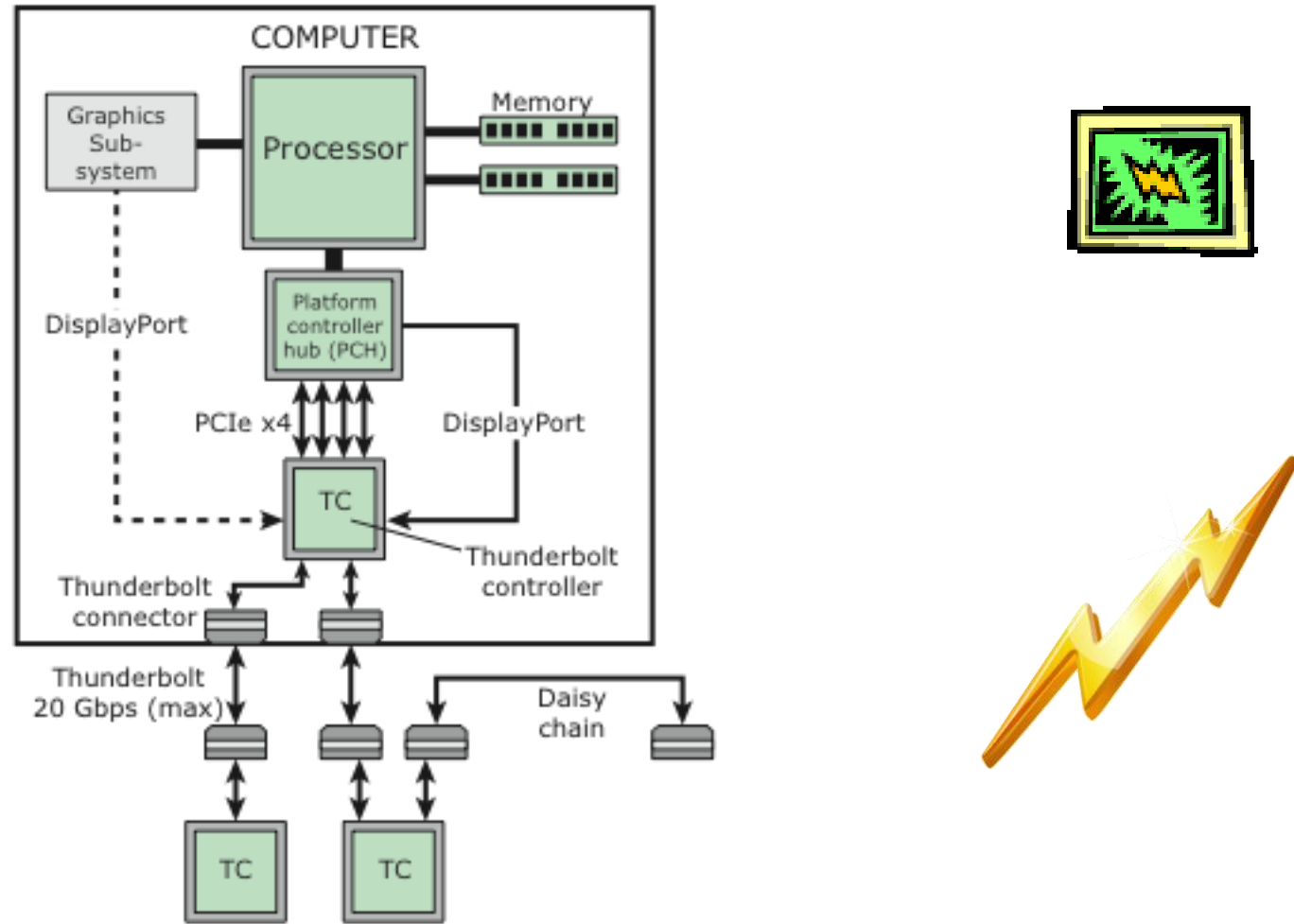




# Thunderbolt



- Most recent and fastest peripheral connection technology to become available for general-purpose use
- Developed by Intel with collaboration from Apple
- The technology combines data, video, audio, and power into a single high-speed connection for peripherals such as hard drives, RAID arrays, video-capture boxes, and network interfaces
- Provides up to 10 Gbps throughput in each direction and up to 10 Watts of power to connected peripherals
- A Thunderbolt-compatible peripheral interface is considerably more complex than a simple USB device
- First generation products are primarily aimed at the professional-consumer market such as audiovisual editors who want to be able to move large volumes of data quickly between storage devices and laptops
- Thunderbolt is a standard feature of Apple's MacBook Pro laptop and iMac desktop computers



**Figure 7.17 Example Computer Configuration with Thunderbolt**

Computer Configuration with Thunderbolt



# Thunderbolt Protocol Layers

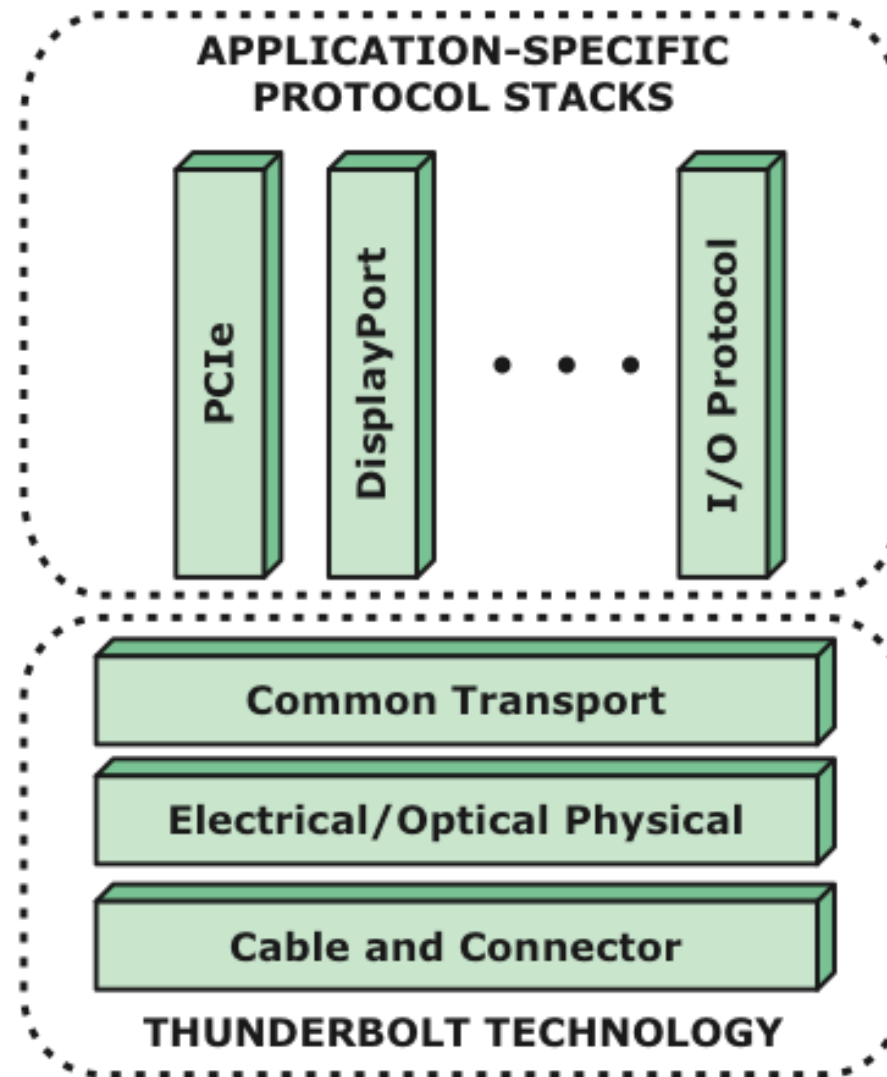
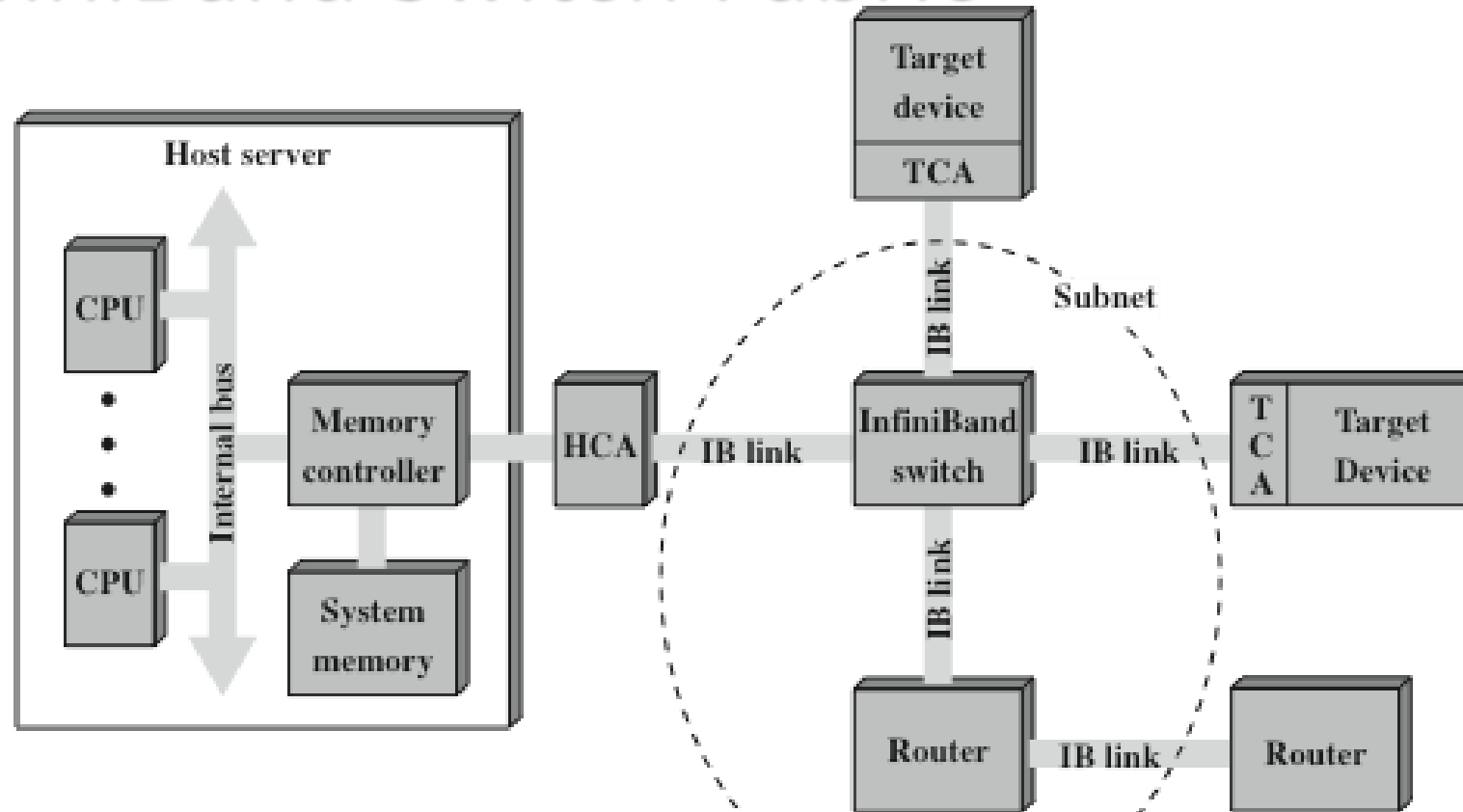


Figure 7.18 Thunderbolt Protocol Layers

# InfiniBand

- Recent I/O specification aimed at the high-end server market
- First version was released in early 2001
- Standard describes an architecture and specifications for data flow among processors and intelligent I/O devices
- Has become a popular interface for storage area networking and other large storage configurations
- Enables servers, remote storage, and other network devices to be attached in a central fabric of switches and links
- The switch-based architecture can connect up to 64,000 servers, storage systems, and networking devices

# InfiniBand Switch Fabric



**IB = InfiniBand**

**HCA = host channel adapter**

**TCA = target channel adapter**

**Figure 7.19 InfiniBand Switch Fabric**



# InfiniBand Operation



- Each physical link between a switch and an attached interface can support up to 16 logical channels, called *virtual lanes*
  - One lane is reserved for fabric management and the other lanes for data transport
- A virtual lane is temporarily dedicated to the transfer of data from one end node to another over the InfiniBand fabric
- The InfiniBand switch maps traffic from an incoming lane to an outgoing lane to route the data between the desired end points
- A layered protocol architecture is used, consisting of four layers:
  - Physical
  - Link
  - Network
  - Transport

+

# Table 7.3 InfiniBand Links and Data Throughput Rates

<b>Link</b>	<b>Signal rate (unidirectional)</b>	<b>Usable capacity (80% of signal rate)</b>	<b>Effective data throughput (send + receive)</b>
1- wide	2.5 Gbps	2 Gbps (250 MBps)	(250 + 250) MBps
4-wide	10 Gbps	8 Gbps (1 GBps)	(1 + 1) GBps
12-wide	30 Gbps	24 Gbps (3 GBps)	(3 + 3) Gbps

# InfiniBand Communication Protocol Stack

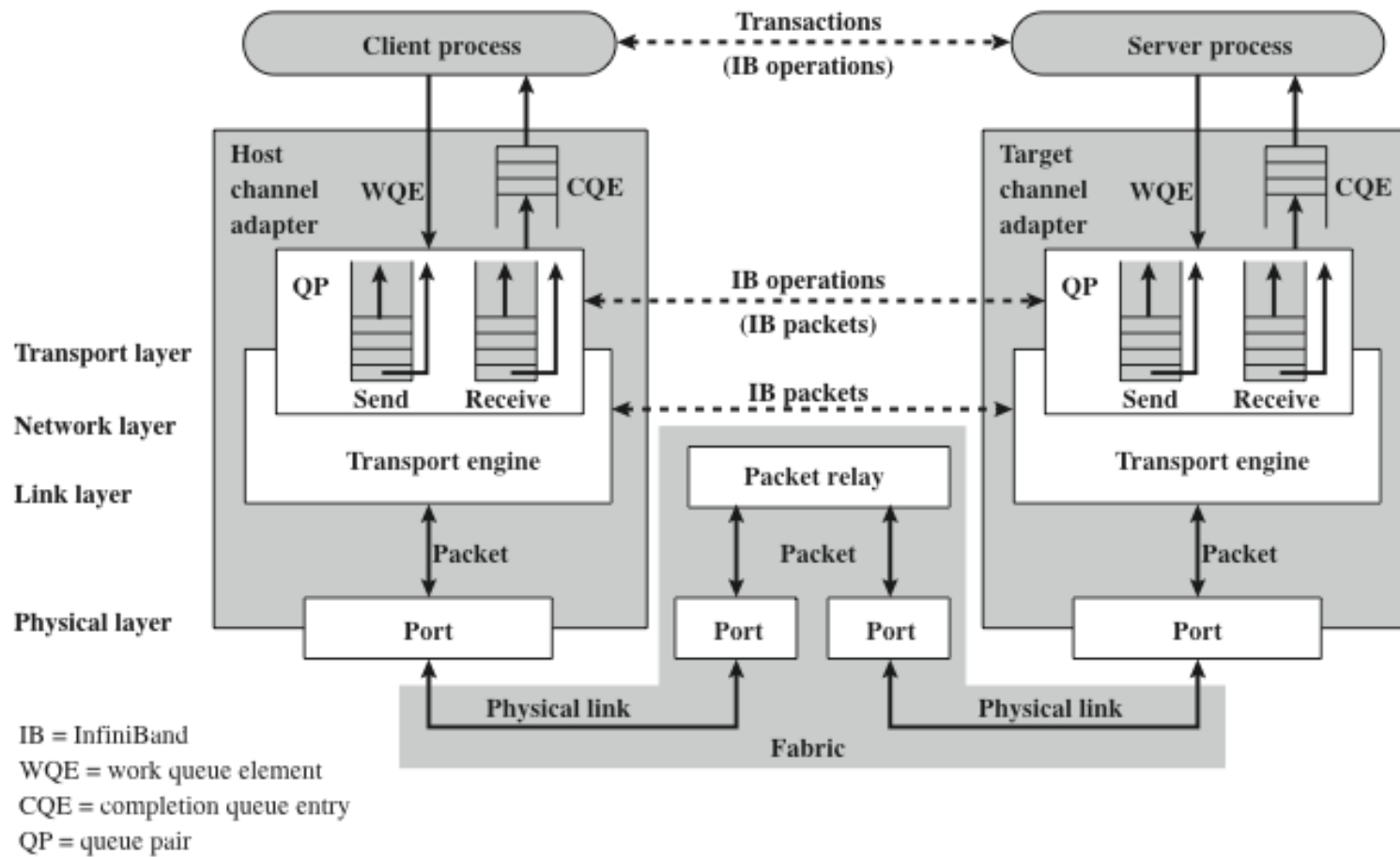
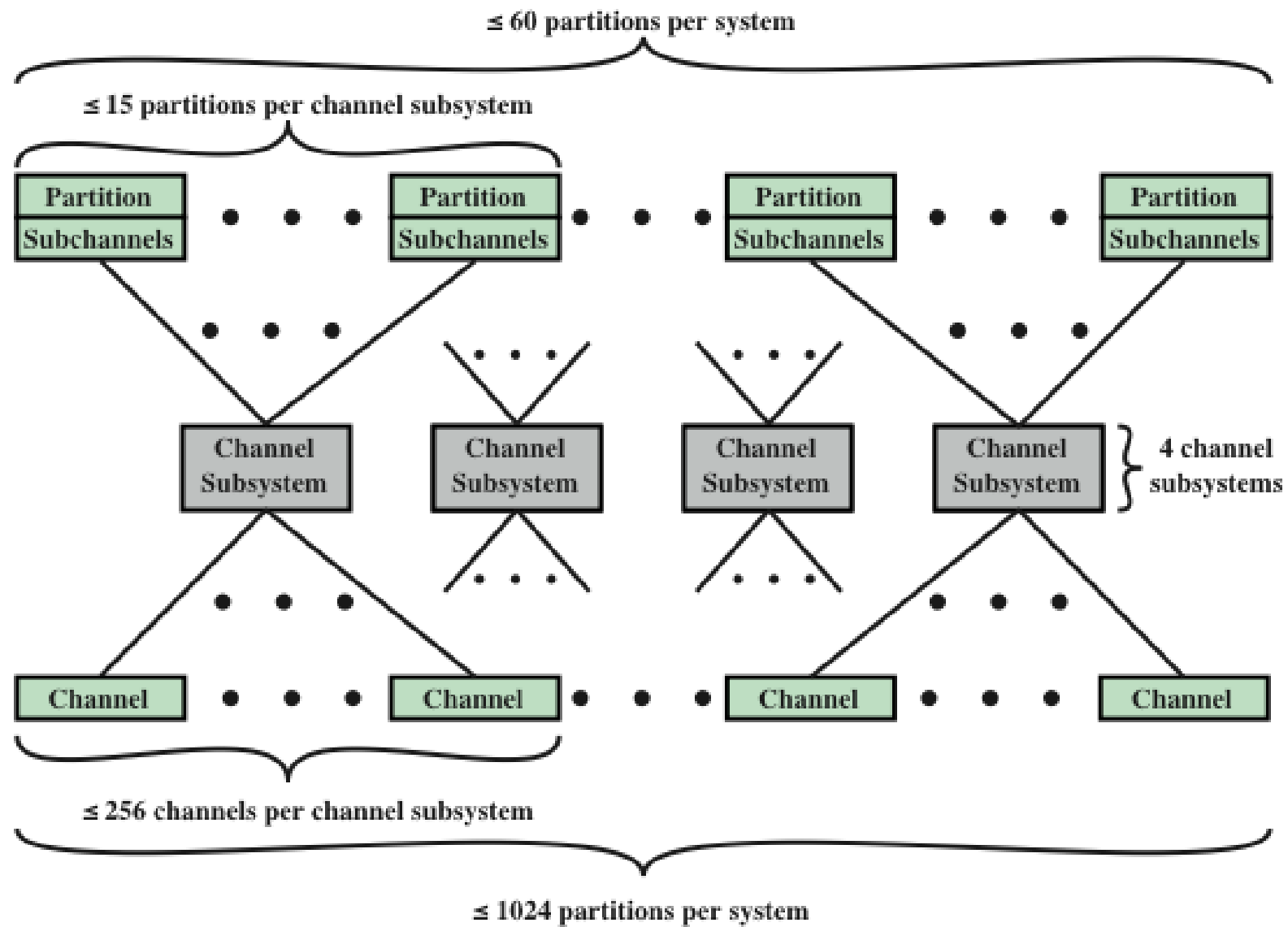


Figure 7.20 Infiniband Communication Protocol Stack

# zEnterprise 196

- Introduced in 2010
- IBM's latest mainframe computer offering
- System is based on the use of the z196 chip
  - 5.2 GHz multi-core chip with four cores
  - Can have a maximum of 24 processor chips (96 cores)
- Has a dedicated I/O subsystem that manages all I/O operations
- Of the 96 core processors, up to 4 of these can be dedicated for I/O use, creating 4 *channel subsystems* (CSS)
- Each CSS is made up of the following elements:
  - System assist processor (SAP)
  - Hardware system area (HSA)
  - Logical partitions
  - Subchannels
  - Channel path
  - Channel



**Figure 7.21 IBM z196 I/O Channel Subsystem Structure**



# I/O System Organization

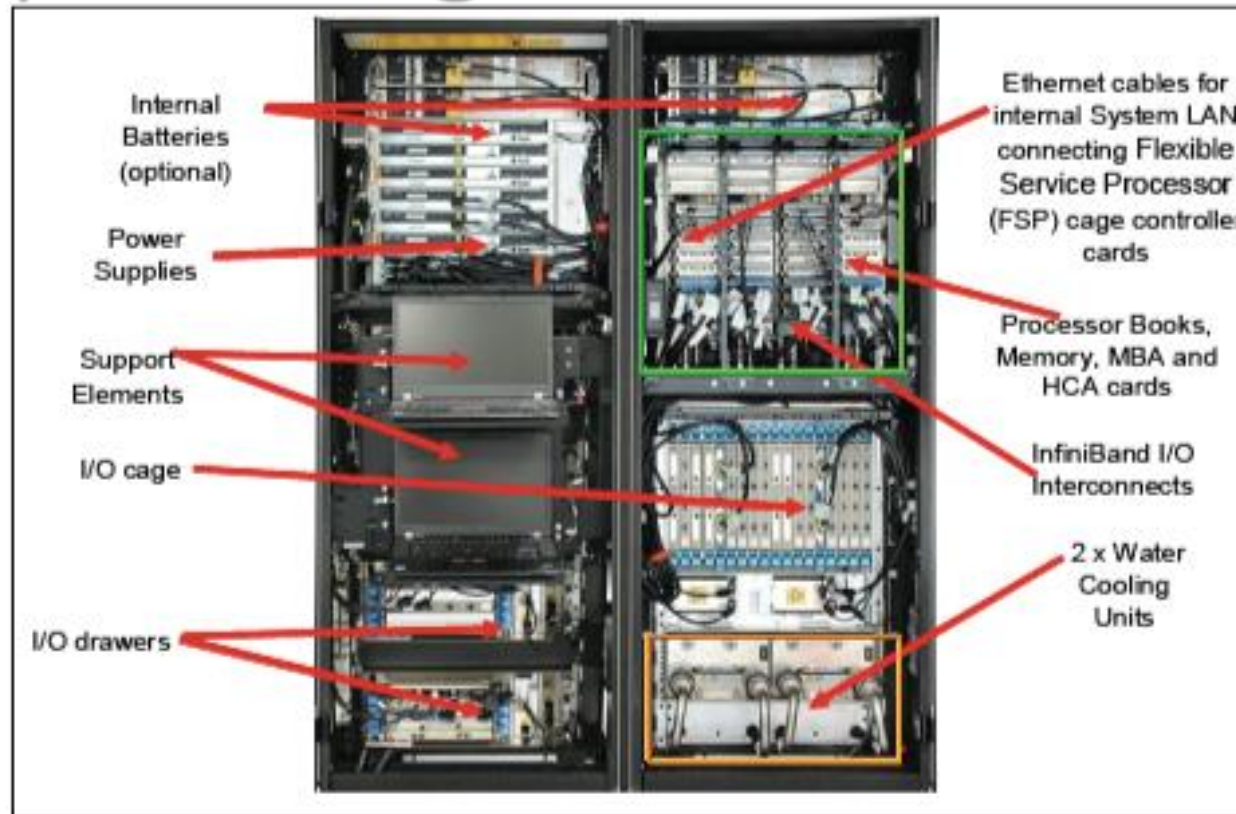


Figure 7.22 IBM z196 I/O Frames — Front View

# IBM z196 I/O System Structure

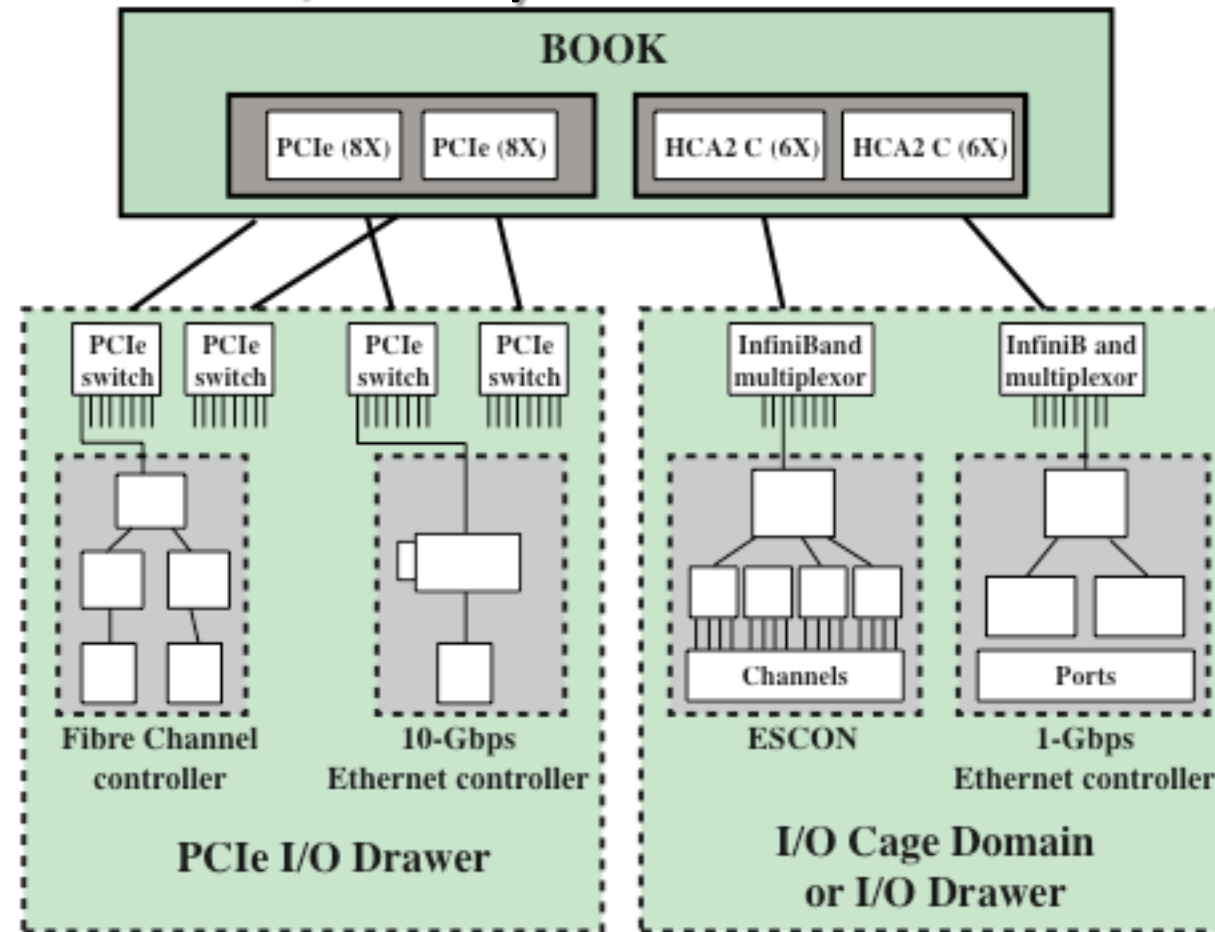


Figure 7.23 IBM z196 I/O System Structure