UNIVERSITI TENAGA NASIONAL

# CHAPTER 12 INPUT OUTPUT

## CSNB153 COMPUTER ORGANIZATION

# Importance of Peripheral Devices

- Interaction between computer and the outside world – make it functional

# I/O Modules

- Contains logic → perform communication function between the peripheral and the bus
- Interface to the system bus/central switch
- Control peripheral devices

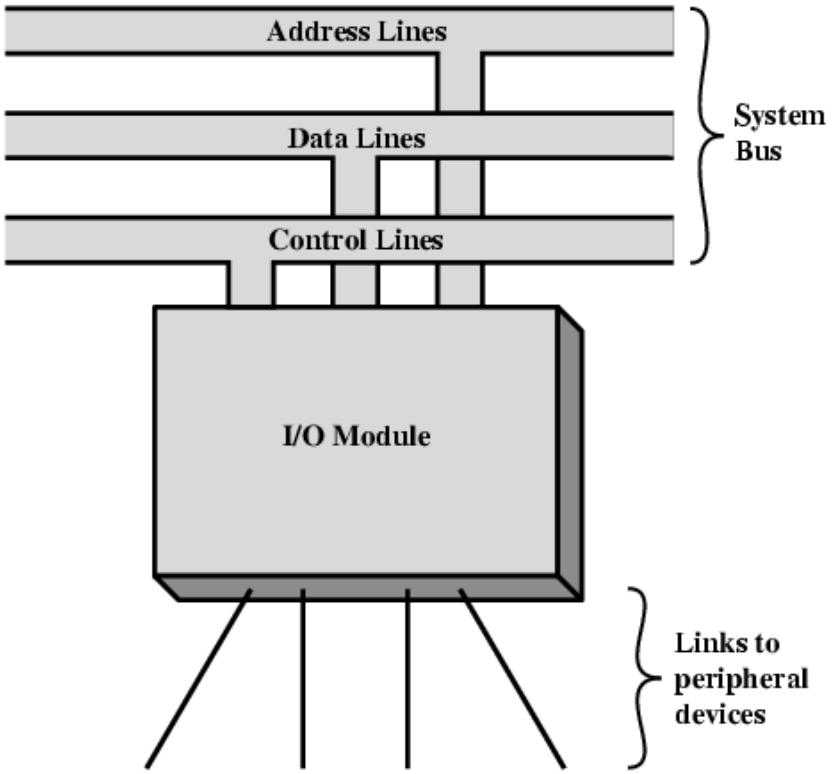
# Importance of I/O Module

- Varieties of peripheral devices → various operation
  - Not practical – burden the processor
  - Hard to control – data transfer rate is slower than processor and memory
  - Use different data formats and word lengths

# Major Function of I/O

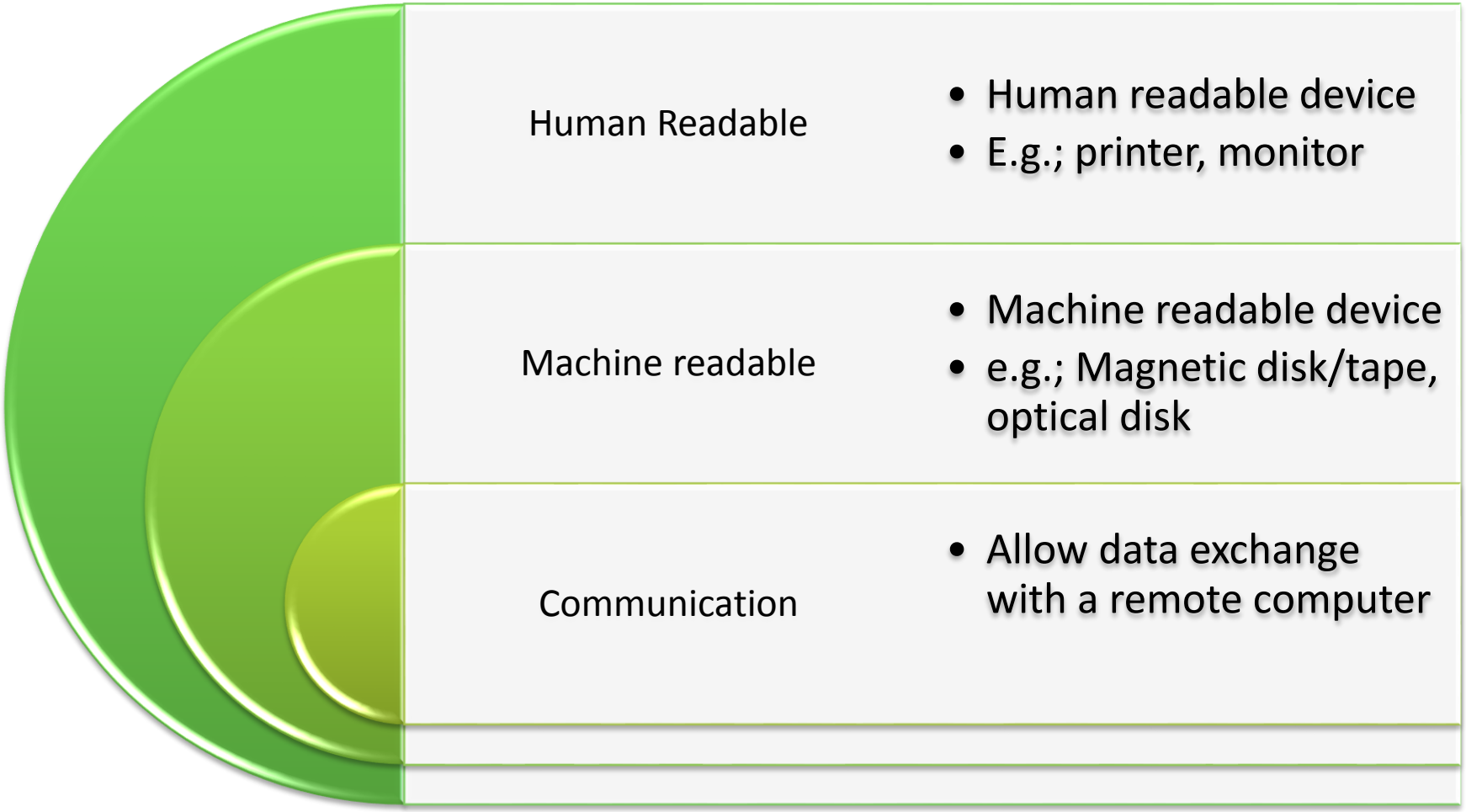
- Interface
  - to the processor and memory via system bus/central switch
  - to one/more peripheral devices

# External Device (ED)

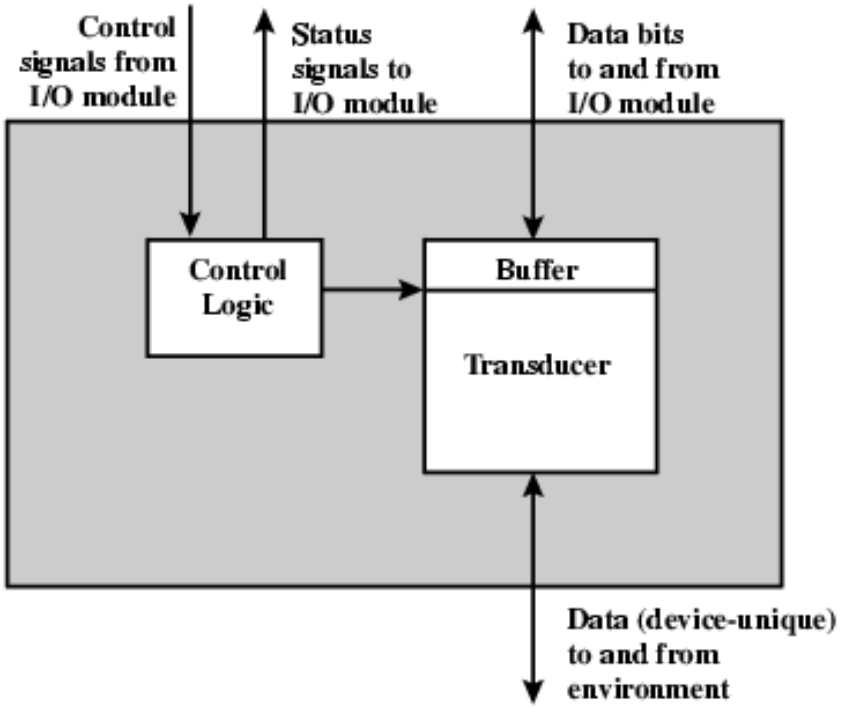


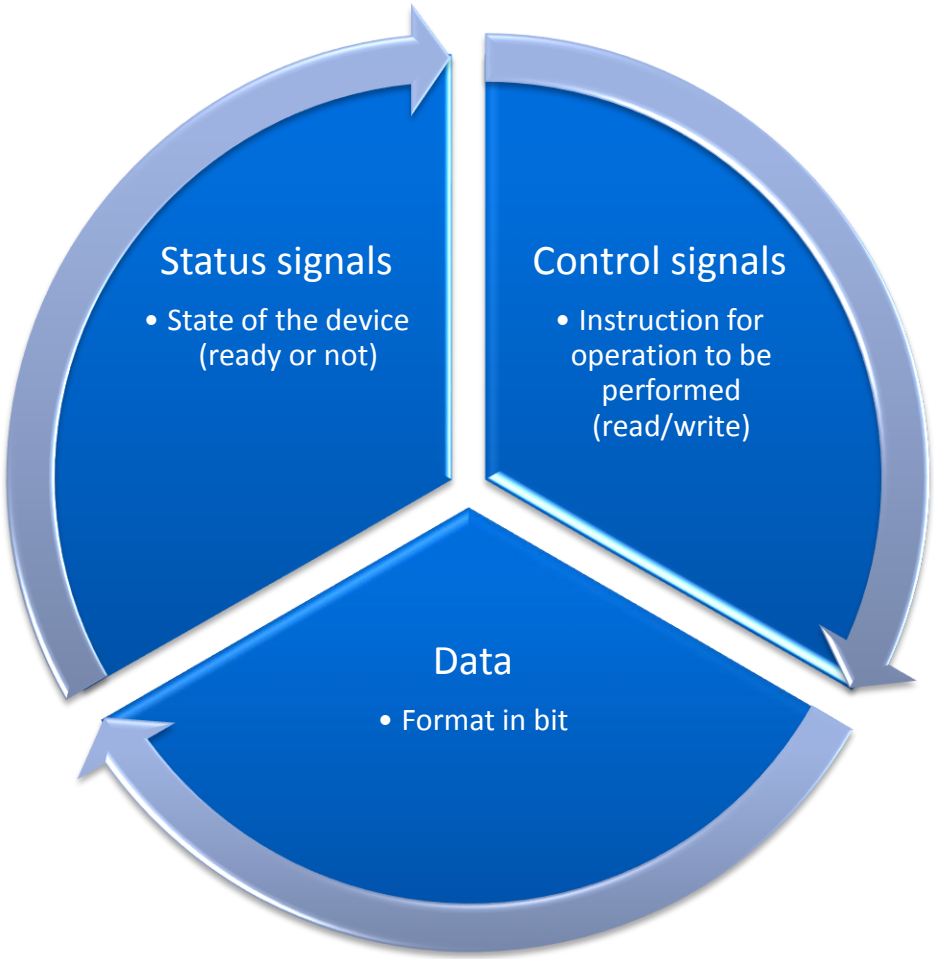

- ED connected to computer by a link to an I/O module
- Use of link;
  - Exchange;
    - Status
    - Control
    - Data

# Classification of ED

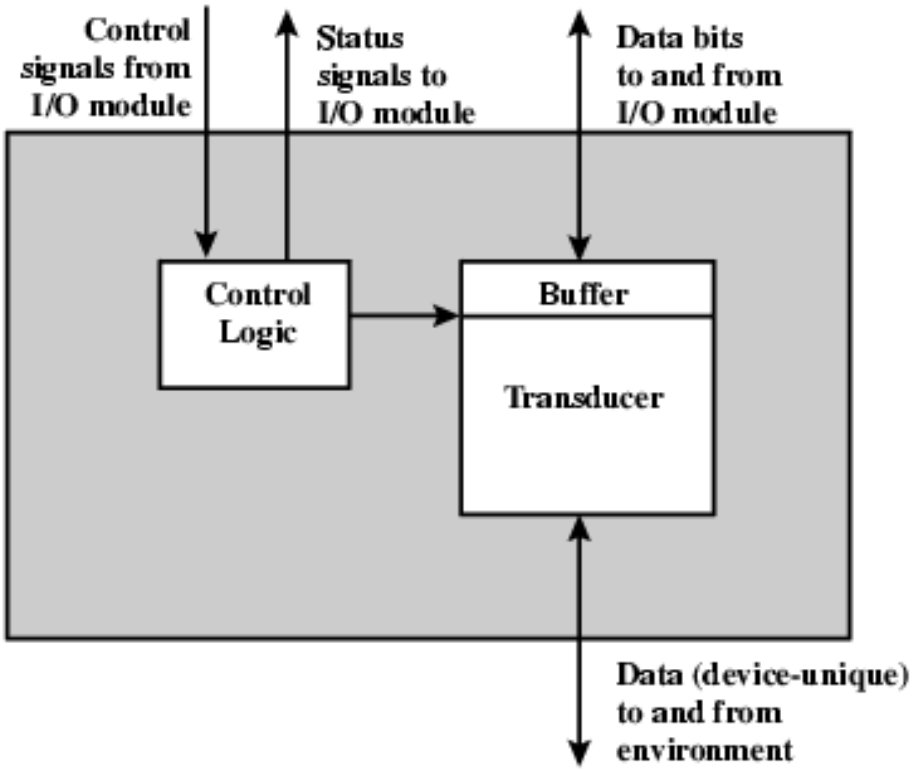


# Interface to I/O Module



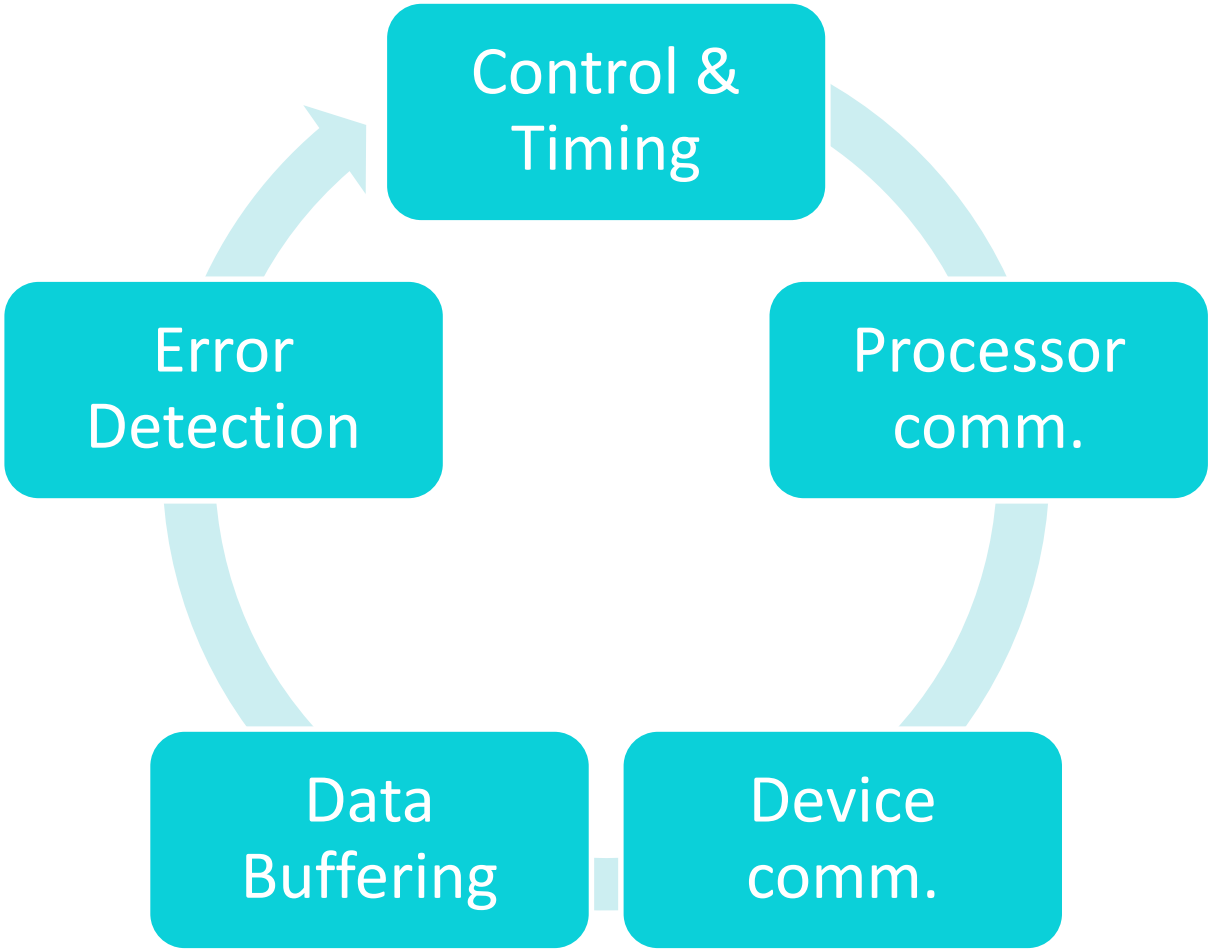



# Interface to I/O Module (Cont.)

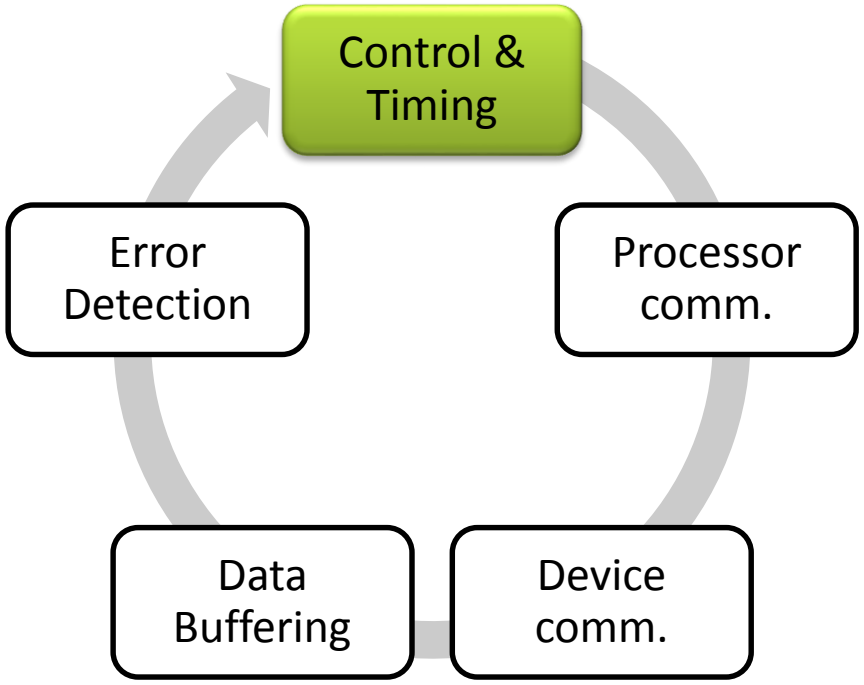



| Term | Description |
|---|---|
| Control logic | controls the device's operation |
| Transducer | converts data;<br>• From electrical to other forms of energy (output)<br>• From other forms to electrical (input) |
| Buffer | related to transducer-hold data temporarily , size 8 to 16 bits |

# Categories of Major Function for I/O Module

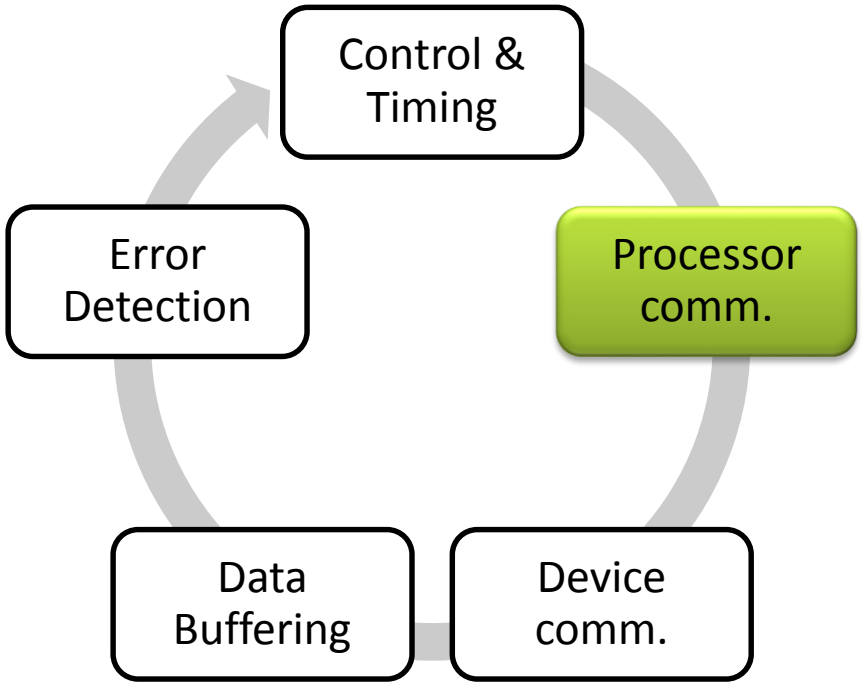


# Categories of Major Function for I/O Module (Cont.)



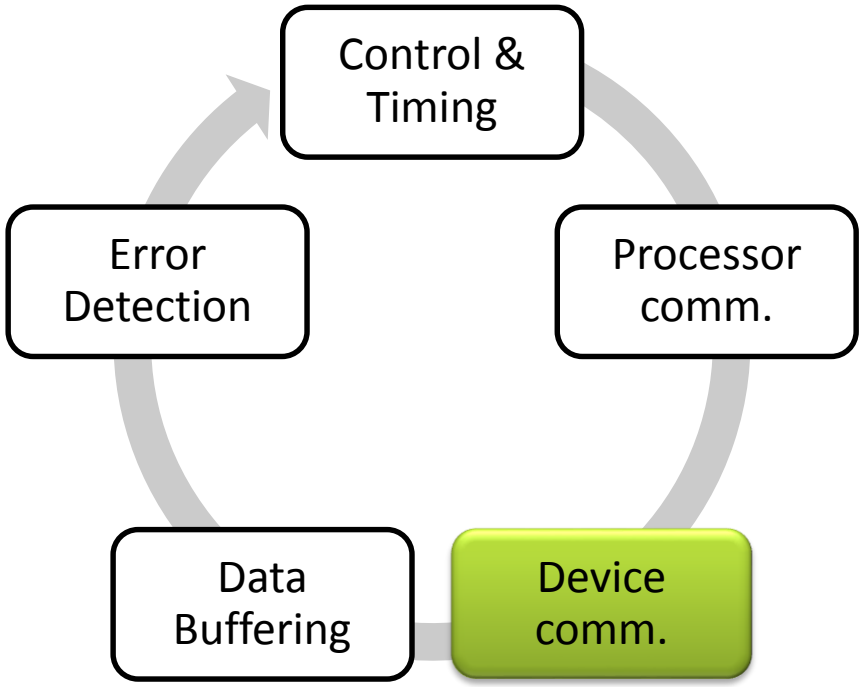

- To coordinate the flow of traffic between internal resources and ED
- E.g. data transfer from ED to CPU
  - Check device status - CPU ask I/O module
  - I/O module returns device status
    - Ready – CPU request data
  - I/O module get data from ED
  - Transfer data

# Categories of Major Function for I/O Module (Cont.)



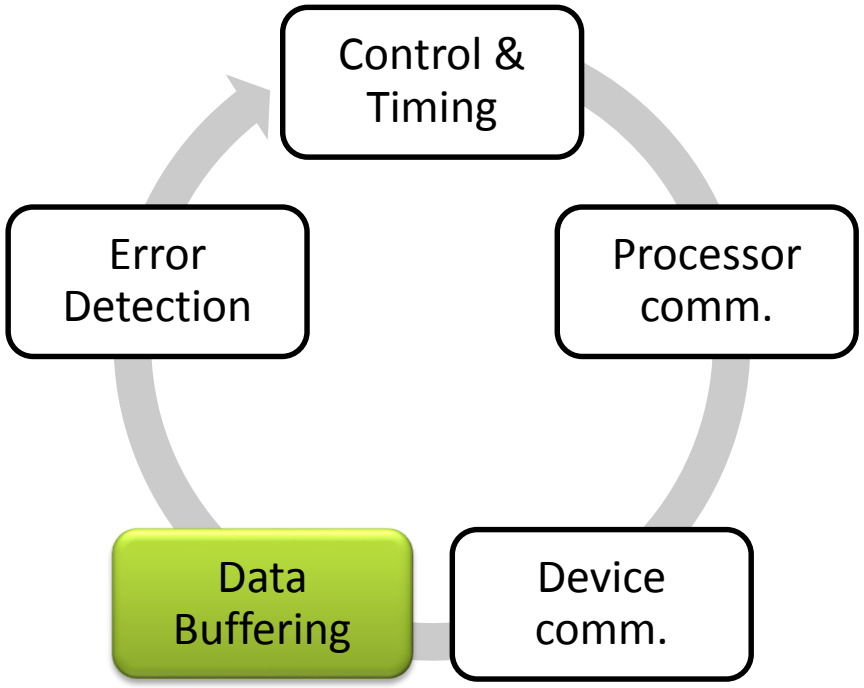

- To communicate between internal resources and ED
- Operations;
  - Command decoding
    - Accept command from CPU send to control bus
  - Data
    - Exchange between CPU and I/O module over data bus
  - Status reporting
    - Status of ED
  - Address recognition
    - Know address for each I/O device

# Categories of Major Function for I/O Module (Cont.)



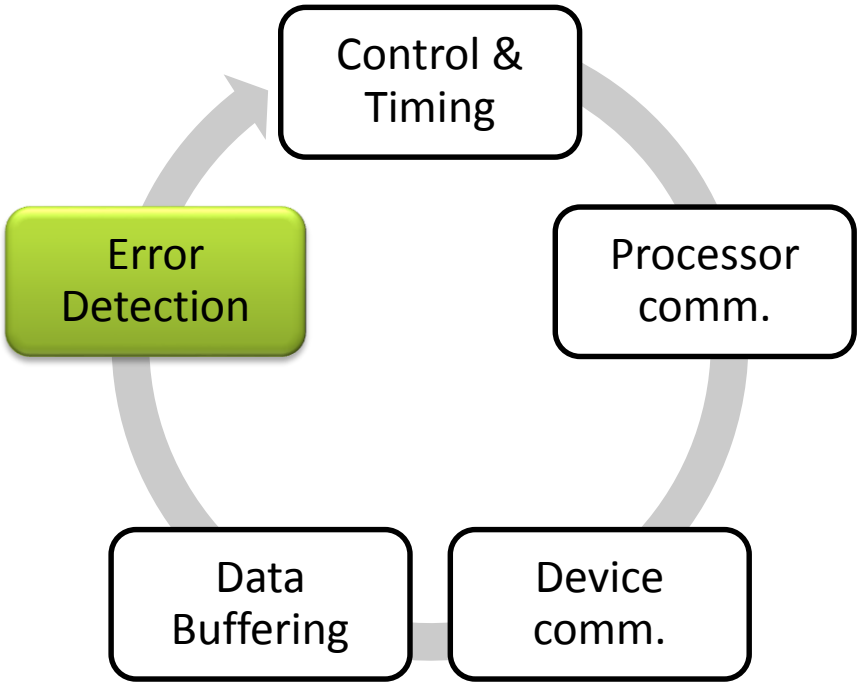

- The I/O module must able to perform DC
- Operations;
  - Command
  - Data
  - Status information

# Categories of Major Function for I/O Module (Cont.)



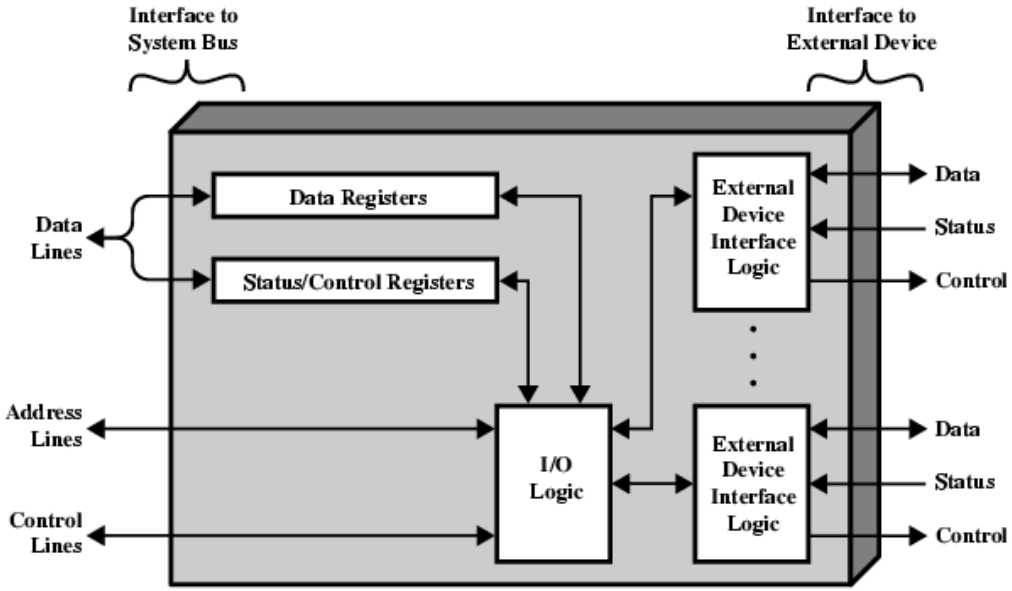

- Data comes from main memory in rapid burst and must be buffered by the I/O module, then send to the device at device's rate

# Categories of Major Function for I/O Module (Cont.)




- I/O module responsible for;
  - error detection
    - Transmission error
  - report errors to the CPU

# I/O Module Structure

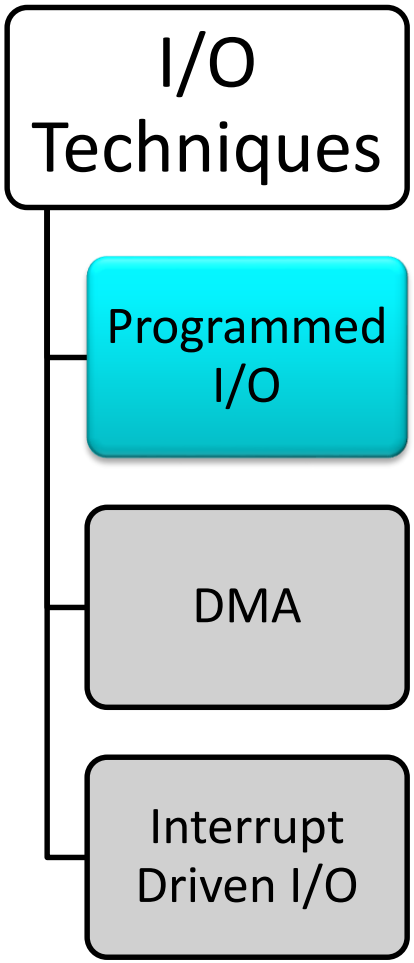



| Term | Description |
|---|---|
| System bus | Connect I/O module to the computer |
| Data registers | Buffer the in/out data into/from I/O module |
| Status/Control registers | Provide register's status |
| Control signal lines | - Allow I/O module logic interacts with CPU<br>- CPU issue command to the IO module |
| Address lines | Know and generate device's address |
| External device interface logic | Device interfaces to the device it's control |
| I/O Logic | Used in primitive I/O module |

# I/O Techniques

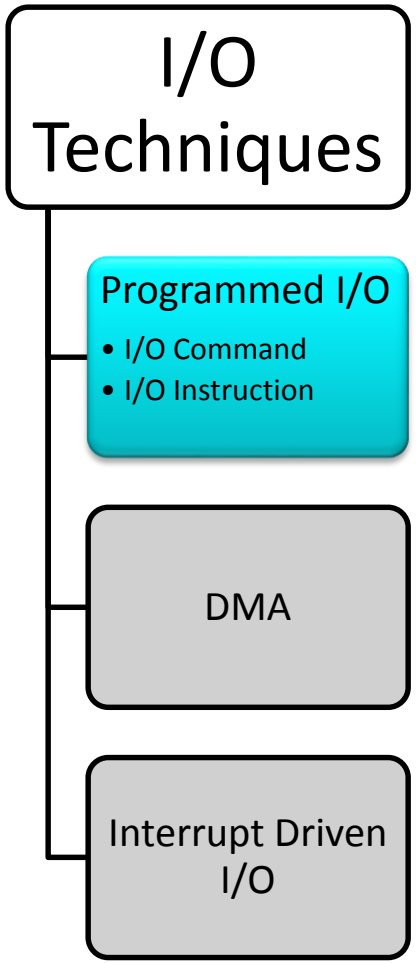
Programmed I/O
Direct Memory Access (DMA)
Interrupt Driven I/O

# Programmed I/O



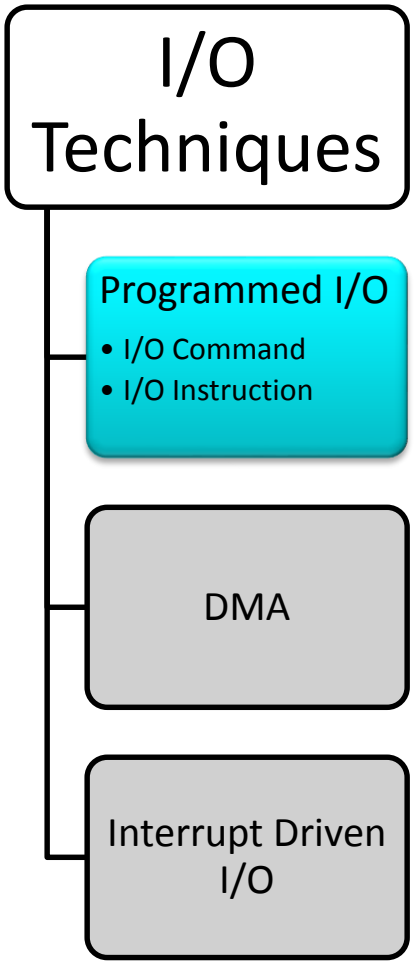

- The processor executes a program that gives it direct control of the I/O operation such as sensing devices status, sending a read/write **command** and transferring the data

# Programmed I/O (Cont.)



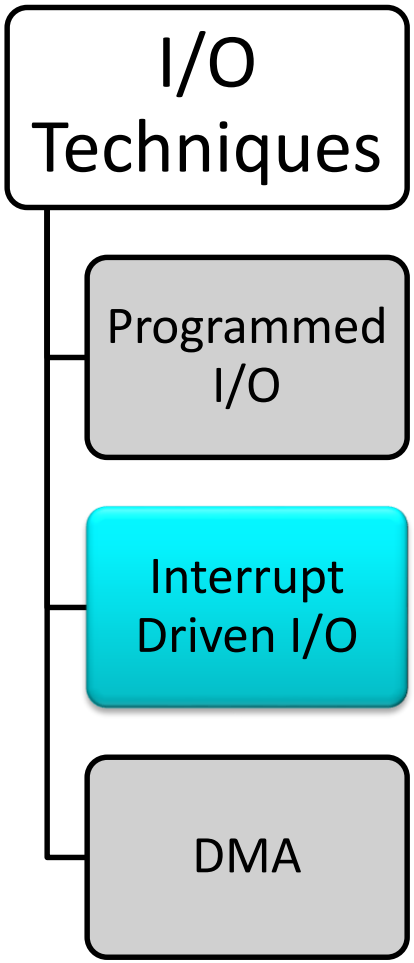

- The CPU issues a command then waits for I/O operations to be complete.
- As the CPU is faster than the I/O module, the problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.
- The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as Polling.
- As a result, the level of the performance of the entire system is severely degraded.

# Programmed I/O (Cont.)



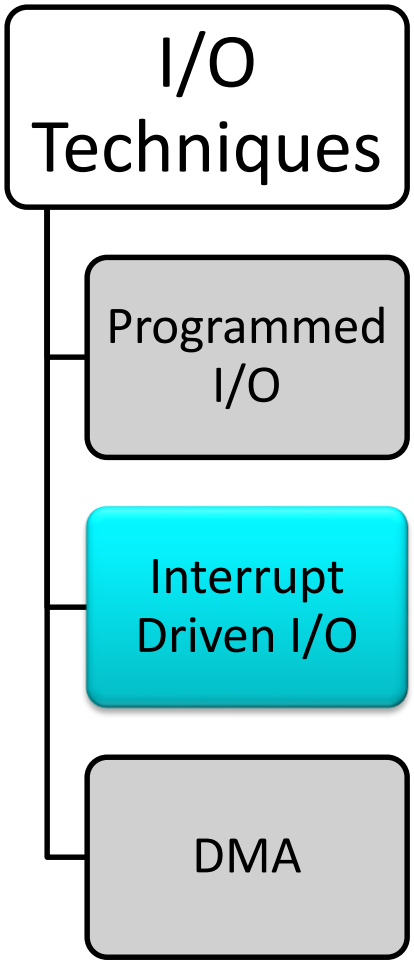

- Programmed I/O basically works in these ways:
  - CPU requests I/O operation
  - I/O module performs operation
  - I/O module sets status bits
  - CPU checks status bits periodically
  - I/O module does not inform CPU directly
  - I/O module does not interrupt CPU
  - CPU may wait or come back later

# Interrupt Driven I/O

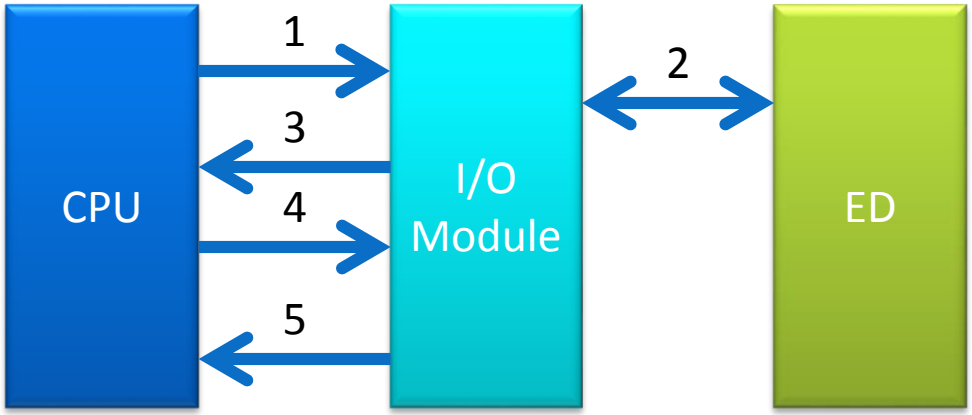



- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

# Interrupt Driven I/O (Cont.)




- Basic operations
  1. CPU issues read command
  2. I/O module gets data from peripheral whilst CPU does other work
  3. I/O module interrupts CPU
  4. CPU requests data
  5. I/O module transfers data



# Interrupt Driven I/O (Cont.) – Design Issue

- Two design issues
  - How does the CPU determine which ED issued the interrupt?
  - If multiple interrupt occured, how does the CPU decide which one to process?

# Interrupt Driven I/O (Cont.) – Design Issue

- Consider device identification. 4 categories;
  - Multiple interrupt lines
  - Software poll
  - Daisy chain
  - Bus arbitration

# Interrupt Driven I/O (Cont.) – Design Issue

- Multiple interrupt lines
  - Straightforward approach
  - Located between CPU and I/O Modules
  - Impractical to dedicate more than a few bus lines or CPU pins to interrupt lines
    - Each line attach to multiple I/O modules

# Interrupt Driven I/O (Cont.) – Design Issue

- Software poll
  - CPU detects interrupt – branches to an interrupt-service routine → job it is to poll each I/O module → determine which module caused the interrupt
  - CPU asks each module in turn – slow (time consuming)

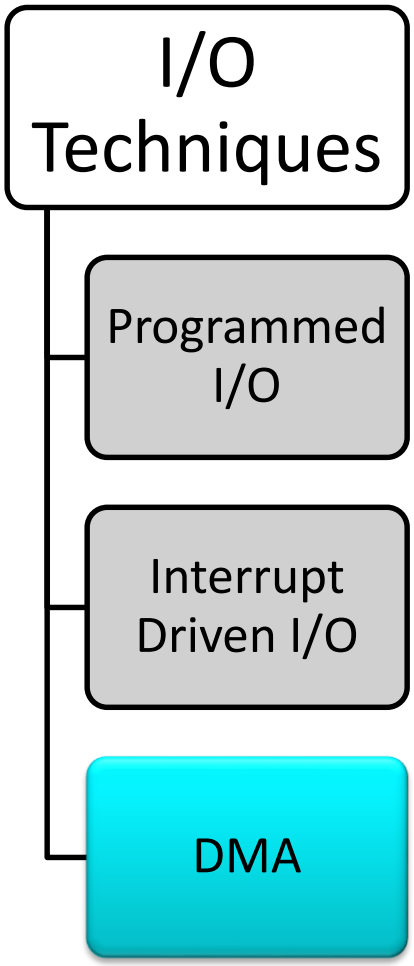
# Interrupt Driven I/O (Cont.) – Design Issue

- Daisy chain
  - Interconnection of computer devices, peripherals, or network nodes in series, one after another
  - All I/O modules share a common interrupt request line
  - The interrupt acknowledge line is daisy chained (hardware poll) through the modules
  - When the CPU senses the interrupt, it send out an interrupt acknowledge
  - This signal propagates via a series of I/O modules till reach a requesting module
  - The requesting model respond and place a word on the data lines

# Interrupt Driven I/O (Cont.) – Design Issue

- Bus arbitration
  - I/O Module must claim the bus before it can raise interrupt
  - Only one module can raise the interrupt at a time
  - When the CPU detects the interrupt, it responds on the interrupt acknowledge line
  - The requesting module then places its vector on the data lines

# Direct Memory Access (DMA)

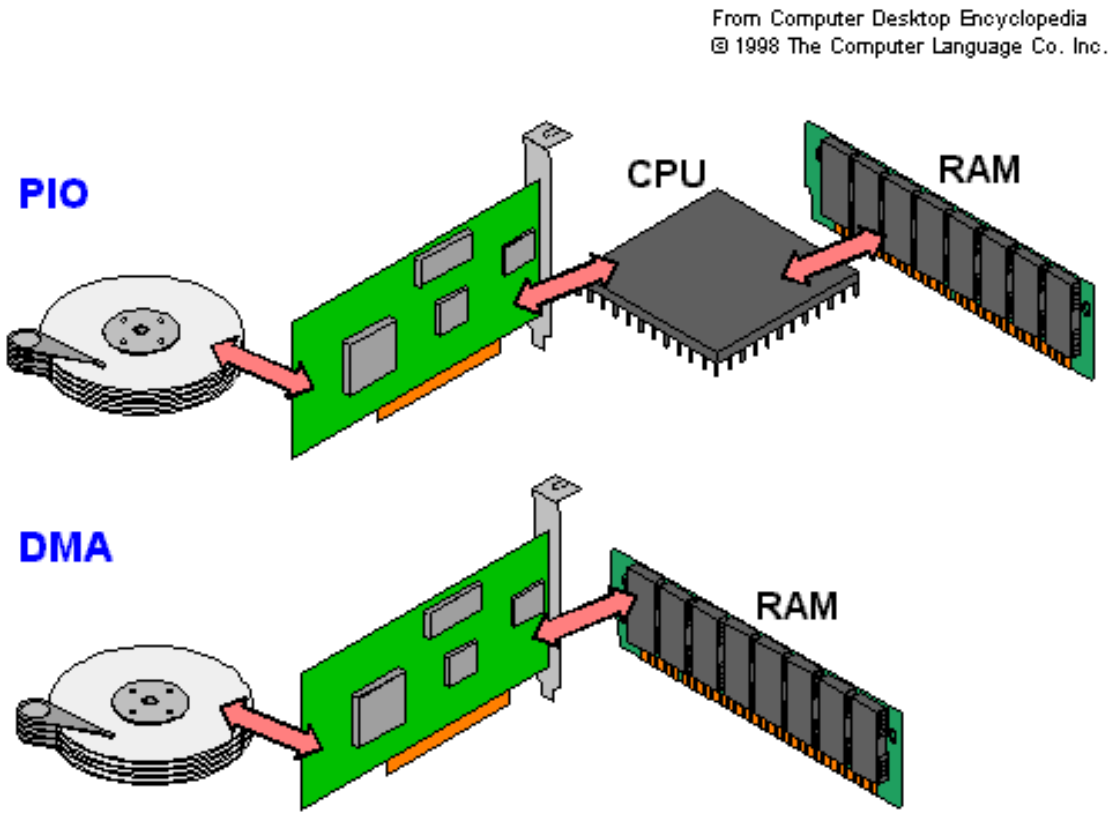



Function

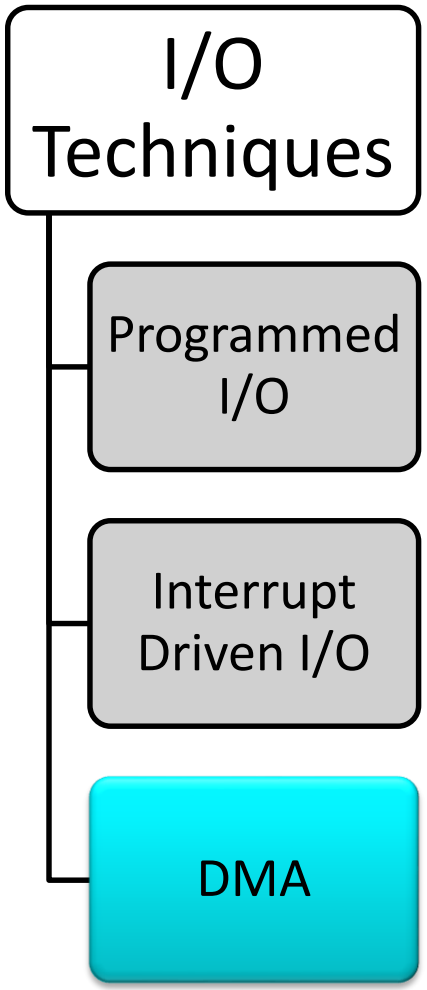
- Additional Module (hardware) on the system bus
- Mimic the CPU – control of the system
- Use system bus for data transferring;
  - when the processor does not need it
  - Force CPU to suspend

# Direct Memory Access (DMA) (Cont.)

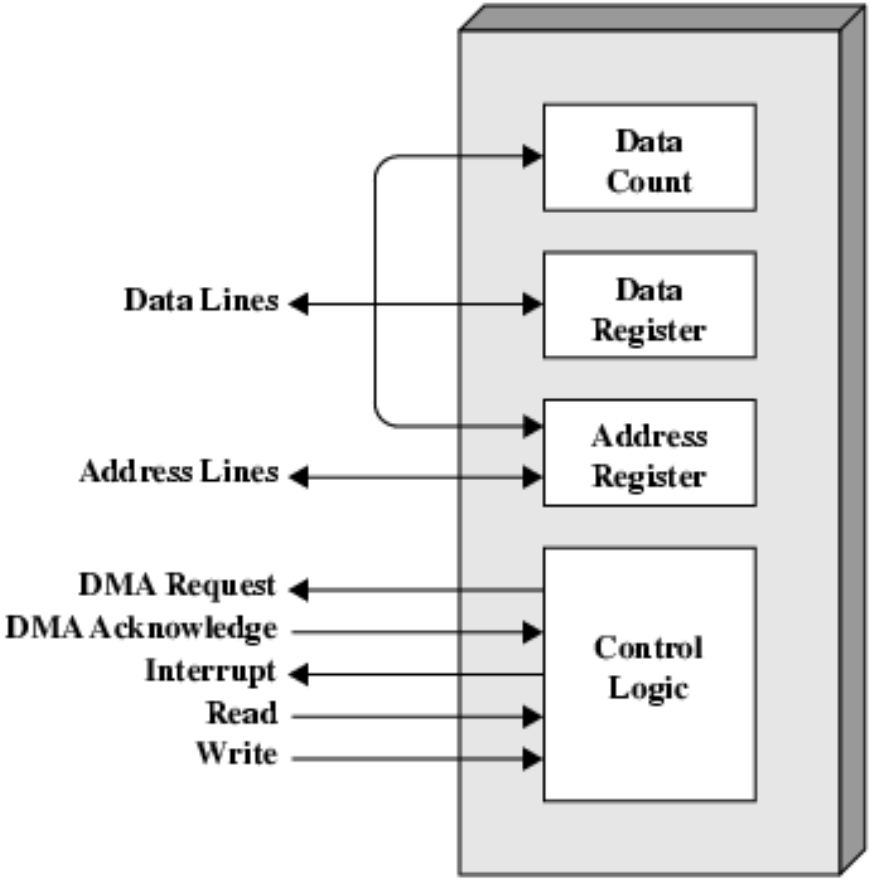
- I/O Techniques
  - Programmed I/O
  - Interrupt Driven I/O
  - DMA



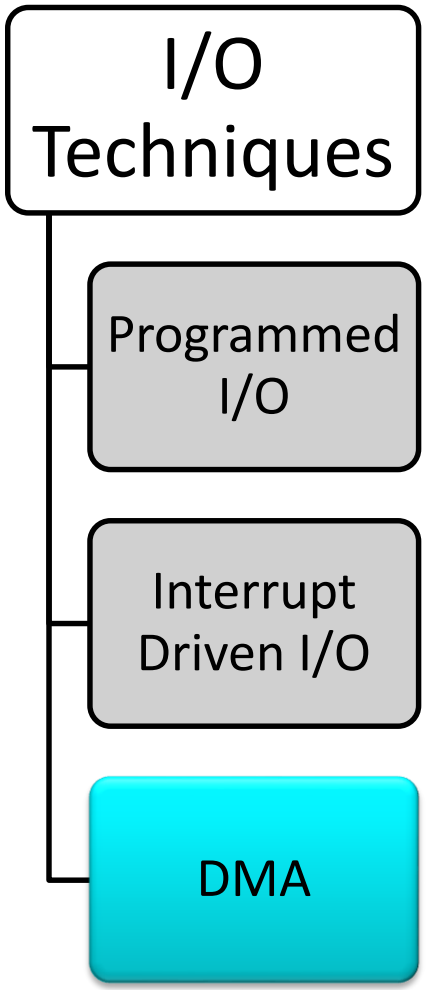
# Direct Memory Access (DMA) (Cont.)




**DMA Block Diagram**



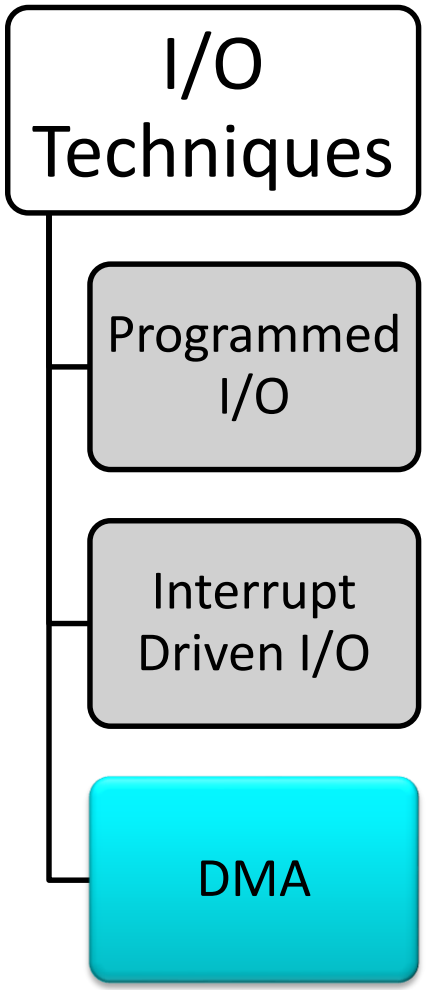
# Direct Memory Access (DMA) (Cont.)




**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

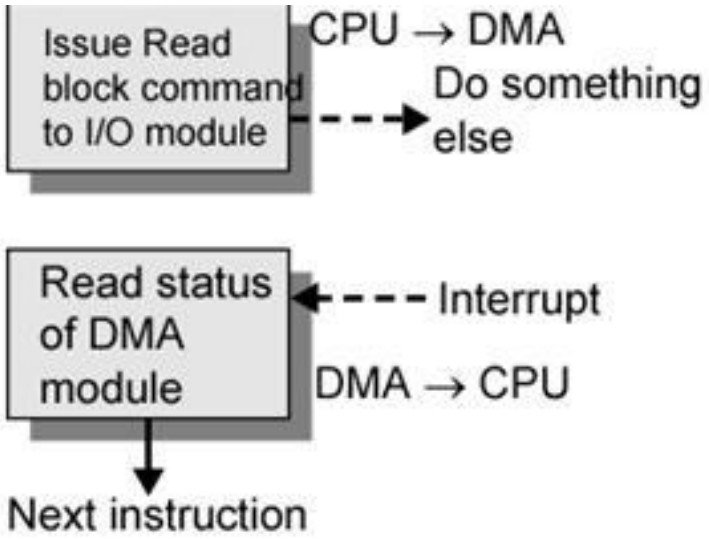
# Direct Memory Access (DMA) (Cont.)




**DMA Transfer – Cycle stealing**

- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Note an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
- Slows down CPU but not as much as CPU doing transfer

# Direct Memory Access (DMA) (Cont.)



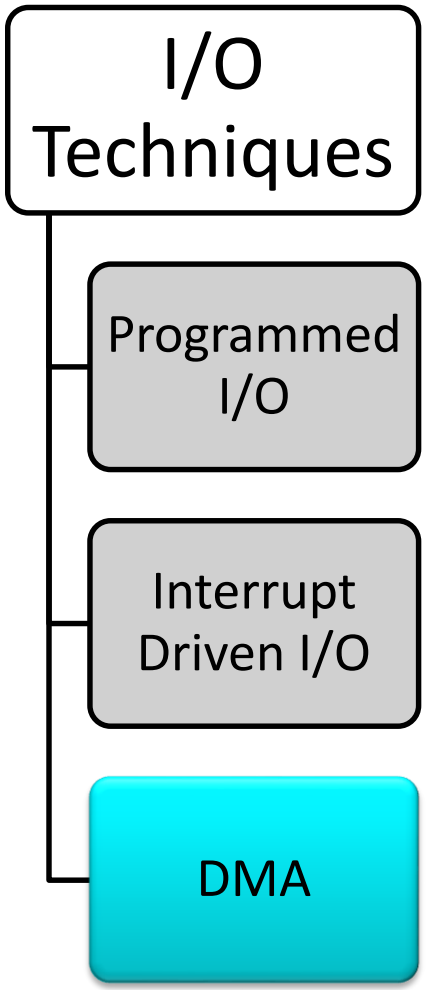

(c) Direct memory access

## DMA Transfer – Cycle stealing (Cont.)

- CPU delegates the I/O operation to DMA module
- The DMA module transfer the entire block of data one at a time directly to or from memory
- When transfer completes - DMA module sends an interrupt signal to the CPU
- CPU only involved at the beginning and end of transmission

- Advantage:
  - DMA is faster than programmed I/O and Interrupt I/O for multiple-word I/O transfer.

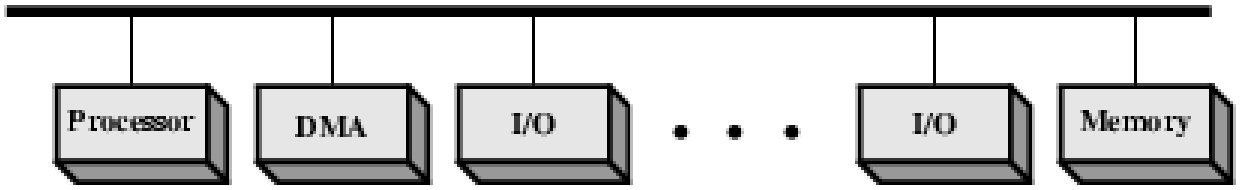
# Direct Memory Access (DMA) (Cont.)




**DMA Configuration**

- Single-bus, detached DMA
- Single-bus, integrated DMA I/O
- I/O bus

# Direct Memory Access (DMA) (Cont.)

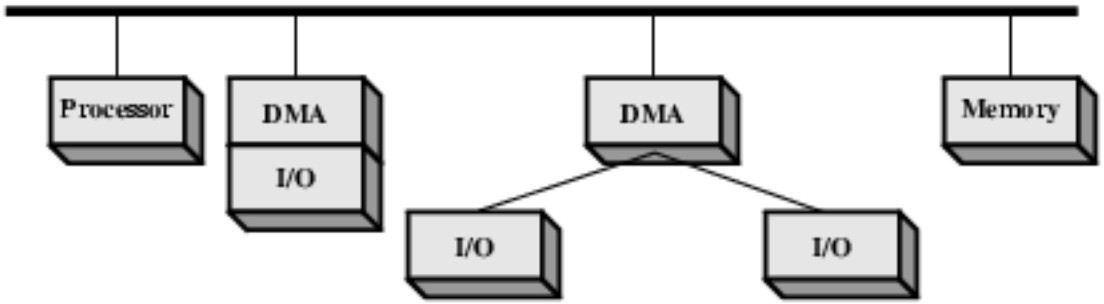



**DMA Configuration –**

**Single-bus, detached DMA**

- All modules share the same system bus
- Each transfer uses bus twice
  - I/O to DMA then DMA to memory

# Direct Memory Access (DMA) (Cont.)



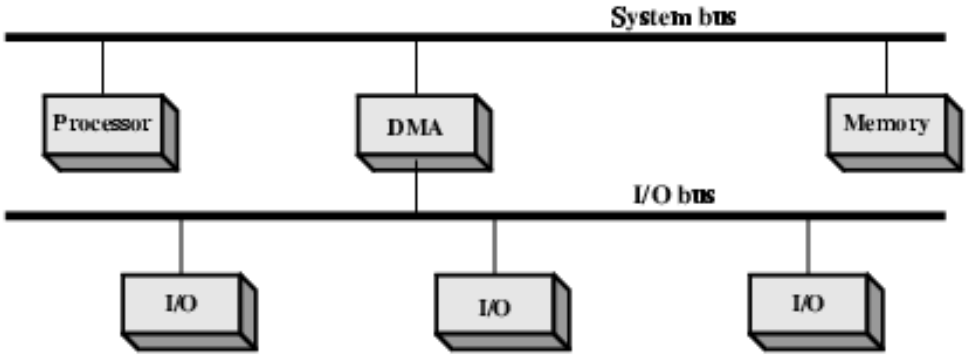

(b) Single-bus, Integrated DMA-I/O

**DMA Configuration –**

**Single-bus, integrated DMA I/O**

- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory

# Direct Memory Access (DMA) (Cont.)




(c) I/O bus

**DMA Configuration –**

**I/O Bus**

- Separate I/O Bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory

# Additional Reference

- William Stallings, Computer Organization and Architecture: Designing for Performance, 8th. Edition, Prentice-Hall Inc., 2010