I/O Management

By: Dr. Upasana Pandey

IMS Engineering College

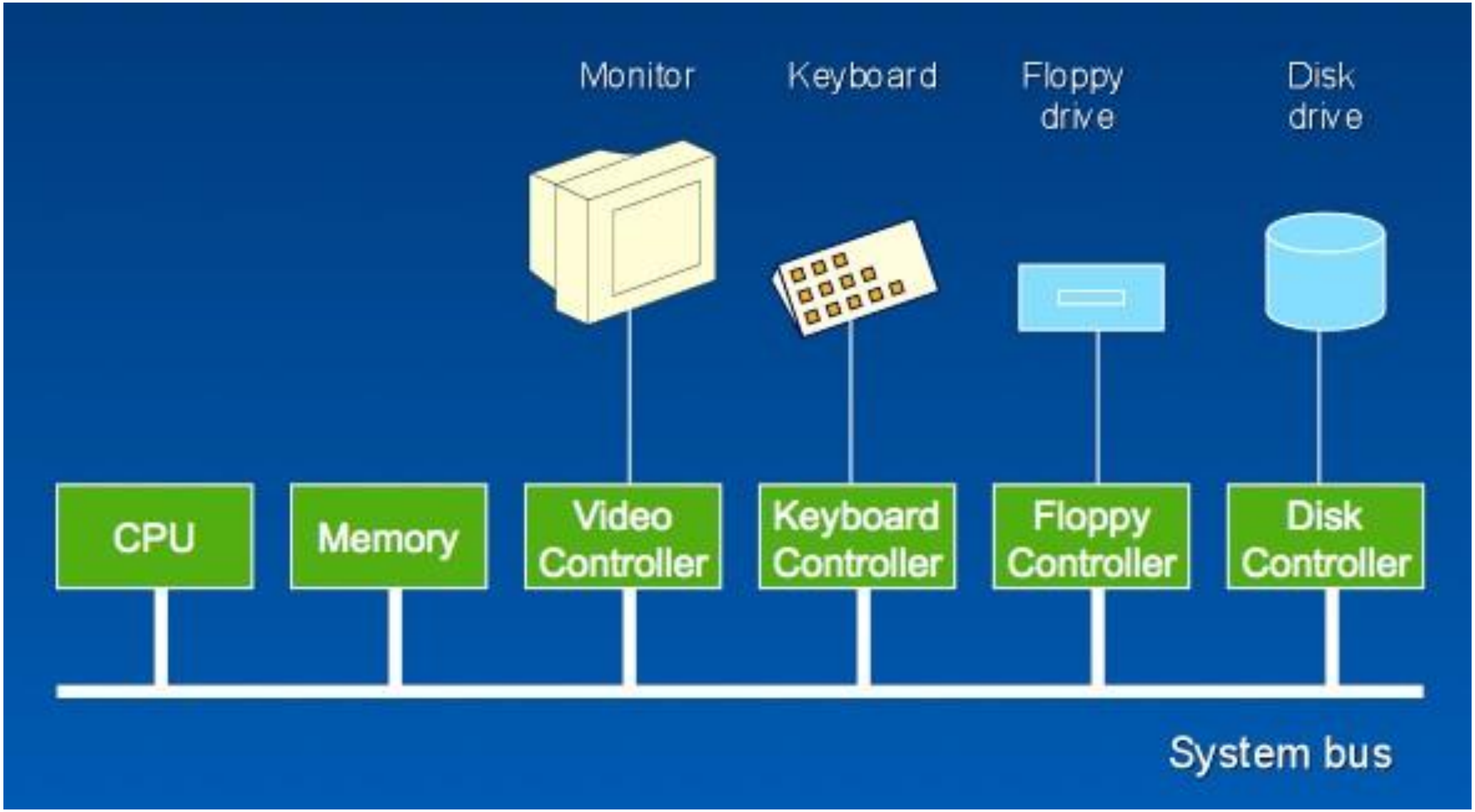
Ghaziabad

Input / Output Devices

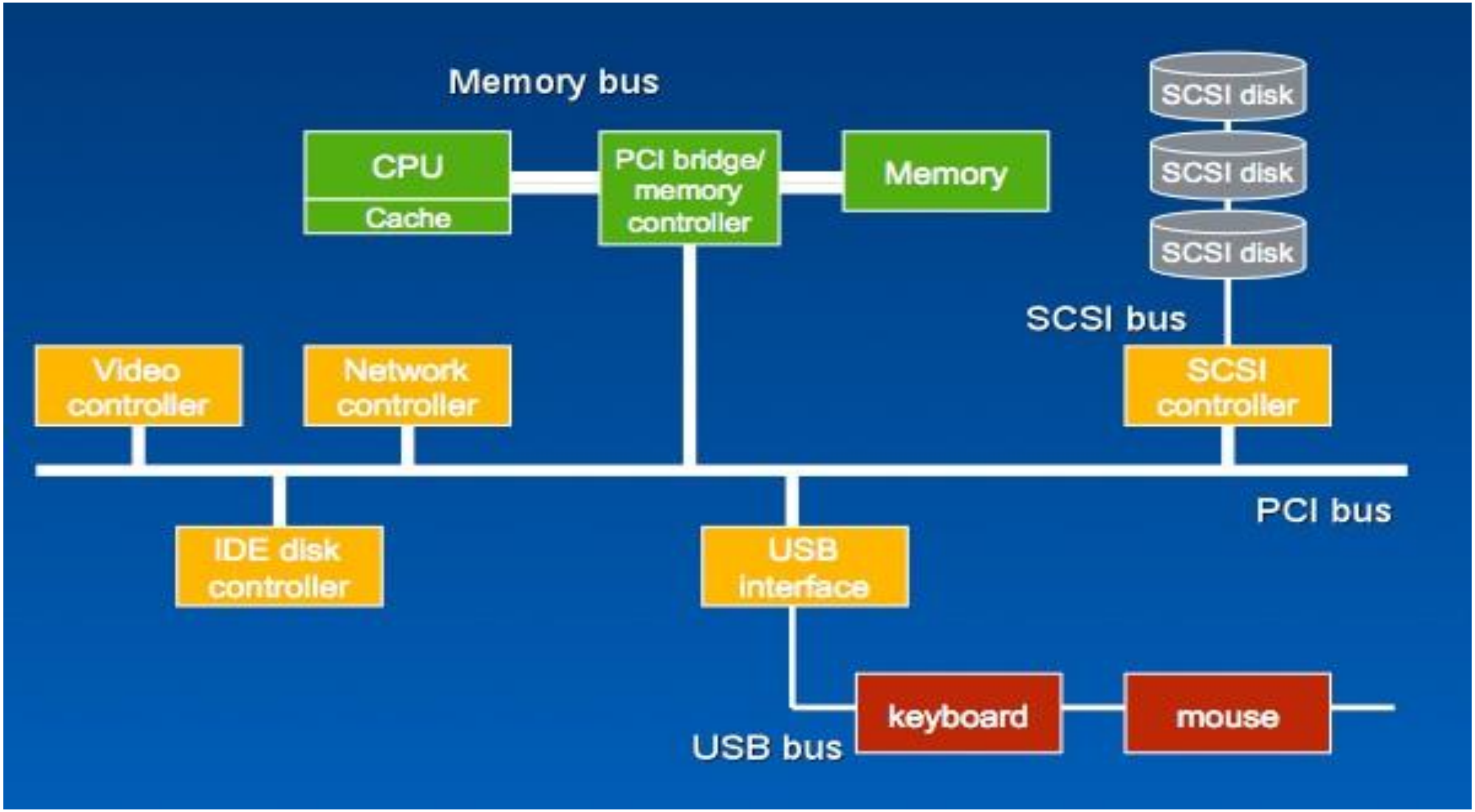
- Hardware devices engaged in I/O can be categorized into:
  - Responsible for interaction with the user
    - Example: Printers, terminals, video display, keyboard, mouse
  - Provisioning of system-local hardware functionality.
    - Disk drives, USB keys, sensors, controllers.
  - Provisioning of communication support
    - Digital line drivers and Modem.
- Devices differ according to multiple factors
  - Data rate, Applications, Complexity of control, Unit of transfer, Data representation, error conditions.

- I/O devices either operate as block device (fixed-size data blocks) or character / stream device (stream of bytes)

I/O Hardware - Single Bus



I/O Hardware- Multiple Buses



I/O Hardware

- An I/O port typically consists of four registers,
  - Status: contains bit that can be read by the host.
  - Control: can be written by the host to start a command or to change the mode of a device.
  - Data-in: is read by the host to get input.
  - Data-out: is written by the host to send output.

I/O Functionality

- Following techniques are used for performing I/O

1. Programmed I/O with Polling
2. Interrupt-driven I/O
3. Direct Memory Access

1. Programmed I/O with Polling

- Producer- Consumer coordinate with the controller and the host by using two bits.
- The controller indicates its state through the busy bit in the status register.
- The controller sets the busy bit when it is busy working and clears the busy bit when it is ready to accept the next command.
- The host signals its wishes via the command-ready bit in the command register.
- The host sets the command-ready bit when a command is available for the controller to execute.

Programmed I/O with Polling (continues…..)

- The host writes output through a port, coordinating with the controller by handshaking as follows:
  1. The host repeatedly reads the busy bit until that bit becomes clear.
  2. The host sets the write bit in the command register and writes a byte into the data-out register.
  3. The host sets the command-ready bit.
  4. When the controller notices that the command-ready bit is set, it sets the busy bit.

Programmed I/O with Polling (continues…..)

5. The controller reads the command register and see the writes command.
6. It reads the data-out register to get the byte, and does the I/O to the device.
7. The controller clears the command-ready bit, clears the error bit in the status register to indicate that the device I/O succeeded and clears the busy bit to indicate that it is finished.

- This loop is repeated for each byte. In step 1, the host is busy-waiting or polling.

2. Interrupt Driven I/O Operation

- Steps for performing an input instruction:
  1. Applications process requests a read operation.
  2. The top half of the device driver queries the status register to determine if the device is idle.
  3. If the device is busy, the driver waits for the device to become idle.
  4. The driver stores an input command into the controller’s command register thereby starting the device.
  5. When the top half of the device driver completes its work, it saves information regarding the operation that it began in the device status table.

Interrupt Driven I/O Operation (Continues….)

6. By that time, the device completes the operation and interrupts the CPU, thereby causing the interrupt handler to run.
7. The interrupt handler determines which device caused the interrupt. It then branches to the device handler for that device.
8. The device handler retrieves the pending I/O status information from the device status table.
9. The device handler copies the contents of the controller data registers into the user process's space.
10. The device handle copies the contents of the controllers data registers into the user process's space.

Inherent Limitations in above approaches

- The I/O transfer rate is limited by the speed.
- The processor is tied up in managing I/O transfer; number of instructions must be executed for each I/O transfer.

Solution: When large volume of the data to be moved, requires more efficient technique named as Direct Memory Access (DMA).

3. Direct Memory Access (DMA)

- A special control unit may be provided to allow transfer of a block of data directly between an eternal device and the main memory without continuous intervention by the processor. This approach is called Direct Memory Access (DMA) .

- DMA is particularly useful on device like disks, where many bytes of information can be transferred in single I/O operations.

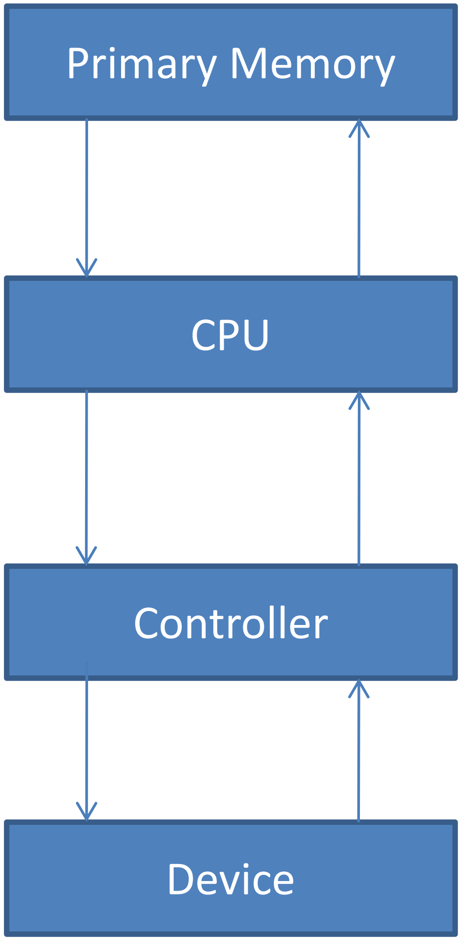
- DMA can be used with either polling or interrupt.

Direct Memory Access (DMA) continuous…..

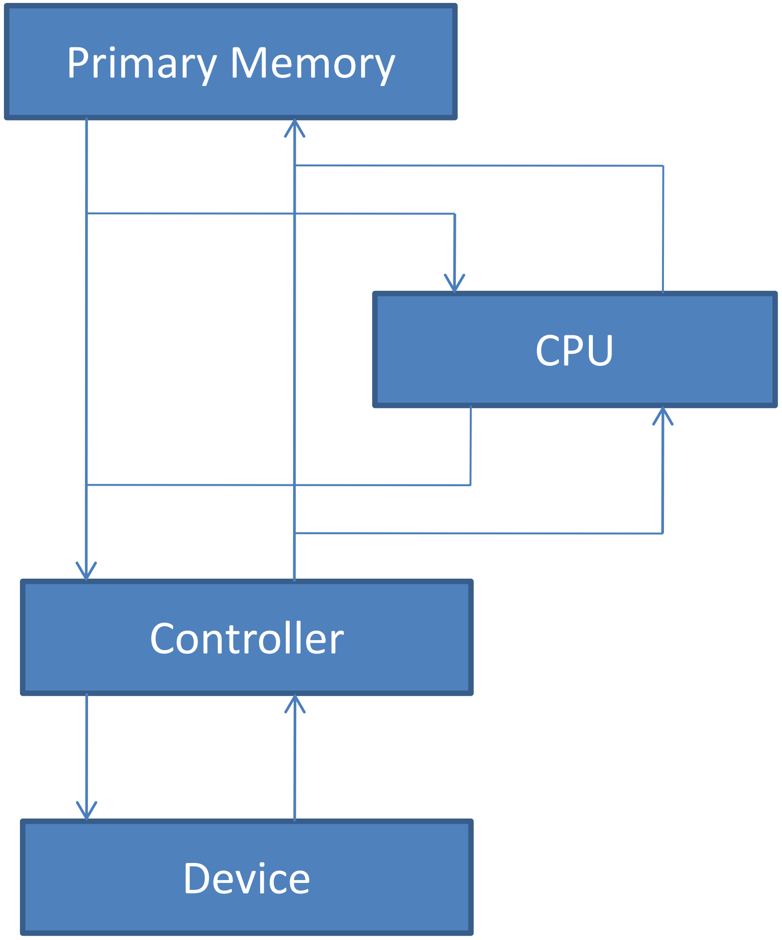
- The technique work as follows:
  - When the processor read or write a block of data, it issues a command to the DMA module by sending the following information:
    - Whether a read or write is requested.
    - The address of the I/O devices.
    - Starting location in memory to read from or write to
    - Number of words to be read or written
  - The processor then continues with other work.
  - The DMA module transfers the entire block of data, one word at a time without going through the processor.
  - When the transfer is complete, the DMA module sends an interrupt signals to the processor.
  - Thus the processor is involved at beginning and end only.

Direct Memory Access (DMA) continue…..

a) Traditional I/O



b) DMA



Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - Character-stream or block
  - Sequential or random-access
  - Synchronous or asynchronous
  - Sharable or dedicated
  - Speed of operation
  - read-write, read only, or write only

Application I/O Interface (continue….)

Sr. No.	Modes	Variation
1	Data Transfer Mode	a. Character b. Block
2	Access Methods	a. Sequential b. Random
3	Transfer Schedule	a. Synchronous b. Asynchronous
4	Sharing	a. Dedicated b. Sharable
5	Device Speed	a. Latency b. Seek time c. Transfer rate d. Delay between operations
6	I/O Operation	a. Read only b. Write only c. Read-Write

Application I/O Interface (Continue....)

- **Character-stream Device**
  - A character stream device transfer bytes one by one.
  - Commands include `get, put`
  - Example- mouse and keyboards
  - Access only serially or sequentially
  - Read the data character by character

- **Block Device**
  - Transfers a block of bytes.
  - Commands include read, write, seek
  - Example- hard disk, floppy disk etc.

Application I/O Interface (Continue….)

- **Network Device**
  - Varying enough from block and character to have own interface
  - Cannot directly transfer data to network devices; instead, they must communicate indirectly by opening a connection to the Kernel’s networking sub- system.

Application I/O Interface (Continue….)

- **Clock and Timers**
  - Provide current time, elapsed time, timer
  - **Programmable interval timer**
    - the hardware to measure elapsed time and to trigger operations.
    - It can be set to wait a certain amount of time and to generate an interrupt it.
  - OS provides interface for user processes to use timers. When timer interrupt, the Kernel signals the requester and reload the timer.

I/O Buffering

- It is a technique by which the device manager can keep slower I/O device busy during times when a process is not requiring I/O operations.

- OS assigns a buffer in the system portion of main memory.

- Types of buffering schemes:
  - Single
  - Double
  - Circular
  - No buffering

I/O Buffering (Continue…)

- **Single Buffer:**
  - Input transfers are made to the system buffer.
  - After transferring, the process moves the block into user space and requests for another block.
  - User process can be processing one block of data while the next block is being read in.
  - OS is able to swap the process out.
  - OS must keep track of the assignment of system buffers to user processes.

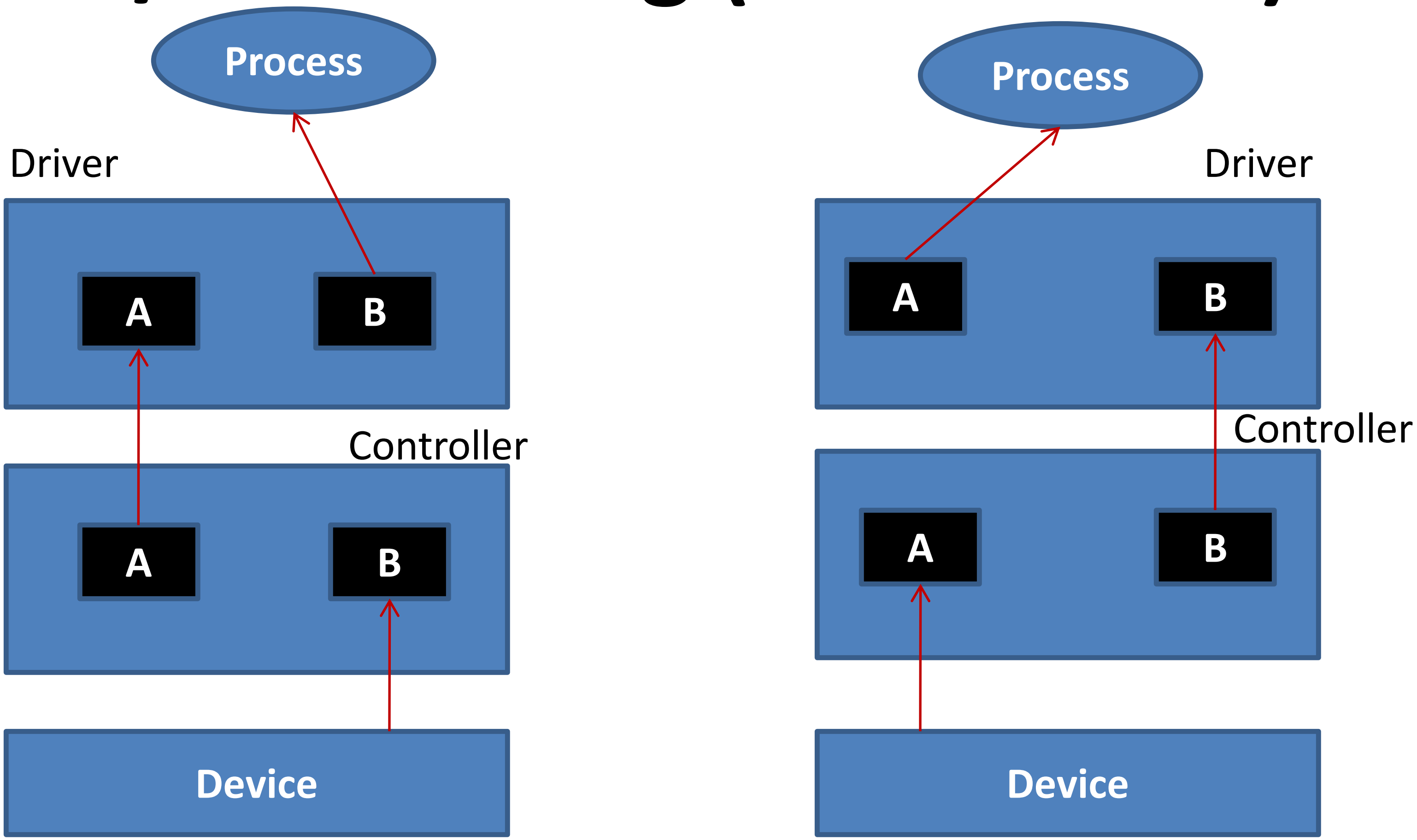
I/O Buffering (Continue…)

Single Buffer

I/O Buffering (Continue…)

- **Double Buffer:**
  - Two buffers in the system
  - One buffer is for the driver or controller to store data while waiting for it to be retrieved by higher level of the hierarchy.
  - Other buffer is to store data from the lower level module.
  - It is also called buffer swapping.

I/O Buffering (Continue…)



Double Buffering

I/O Buffering (Continue…)

- **Circular Buffer:**
  - More than two buffers are used.
  - The producer cannot pass the consumer because it would overwrite buffers before they had been consumed.
  - The producer can only fill up to buffer j-1 while data in buffer j is waiting to be consumed.

Thank You