Chapter 7

Input Output Organization

Input/Output Problems

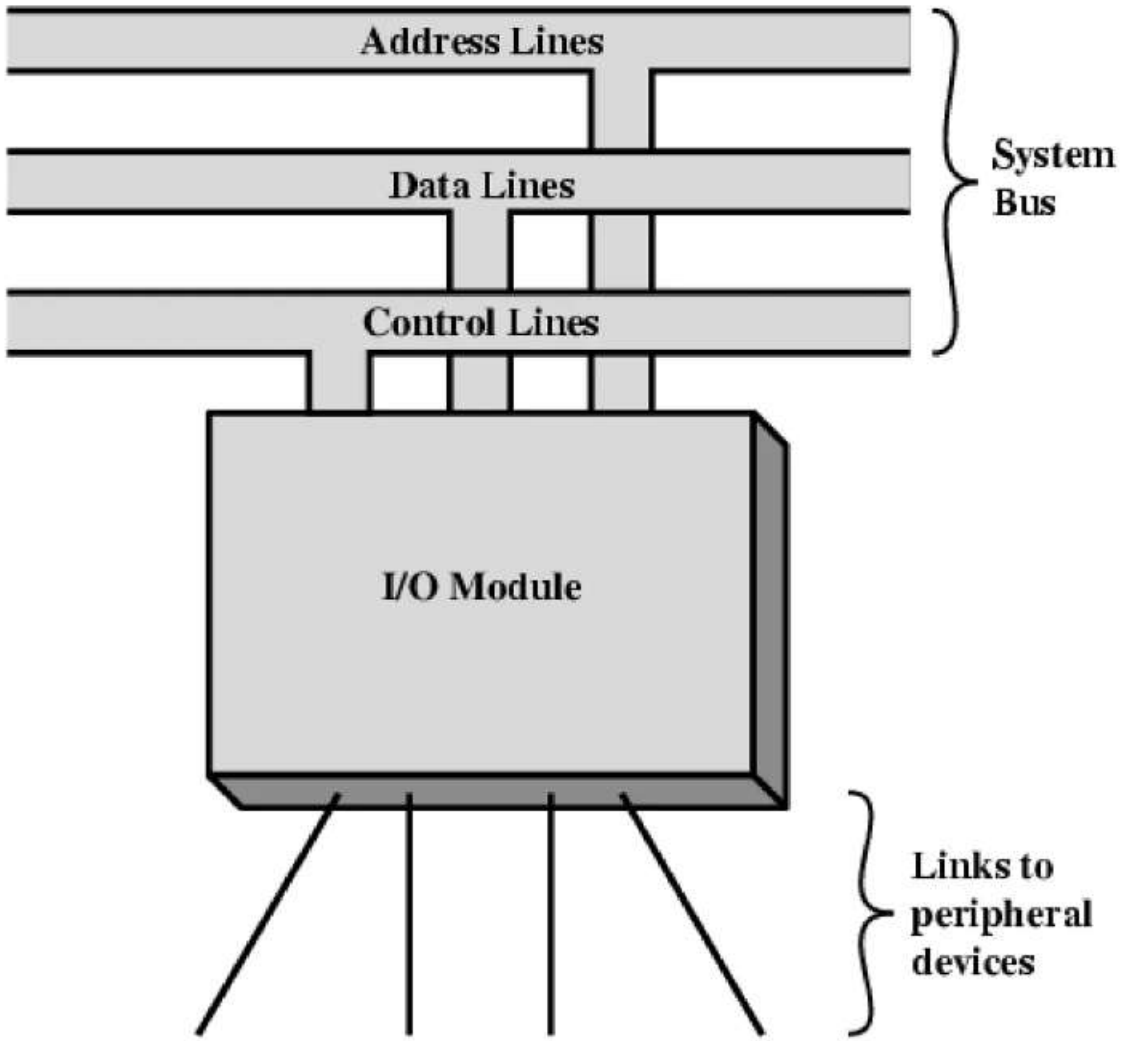
- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and RAM
- Need I/O modules

Input/Output Module

- Interface to CPU and Memory
- Interface to one or more peripherals

Generic Model of I/O Module

- Variety of peripherals with different methods of operations, so it would be practically impossible to define methods for every peripheral.
- Different peripherals have different data transfer rate, different word length.



External Devices

- Human readable
  - —Screen, printer, keyboard
- Machine readable
  - —Monitoring and control Magnetic tapes, Magnetic disks
- Communication
  - —Modem
  - —Network Interface Card (NIC)

External Device Block Diagram

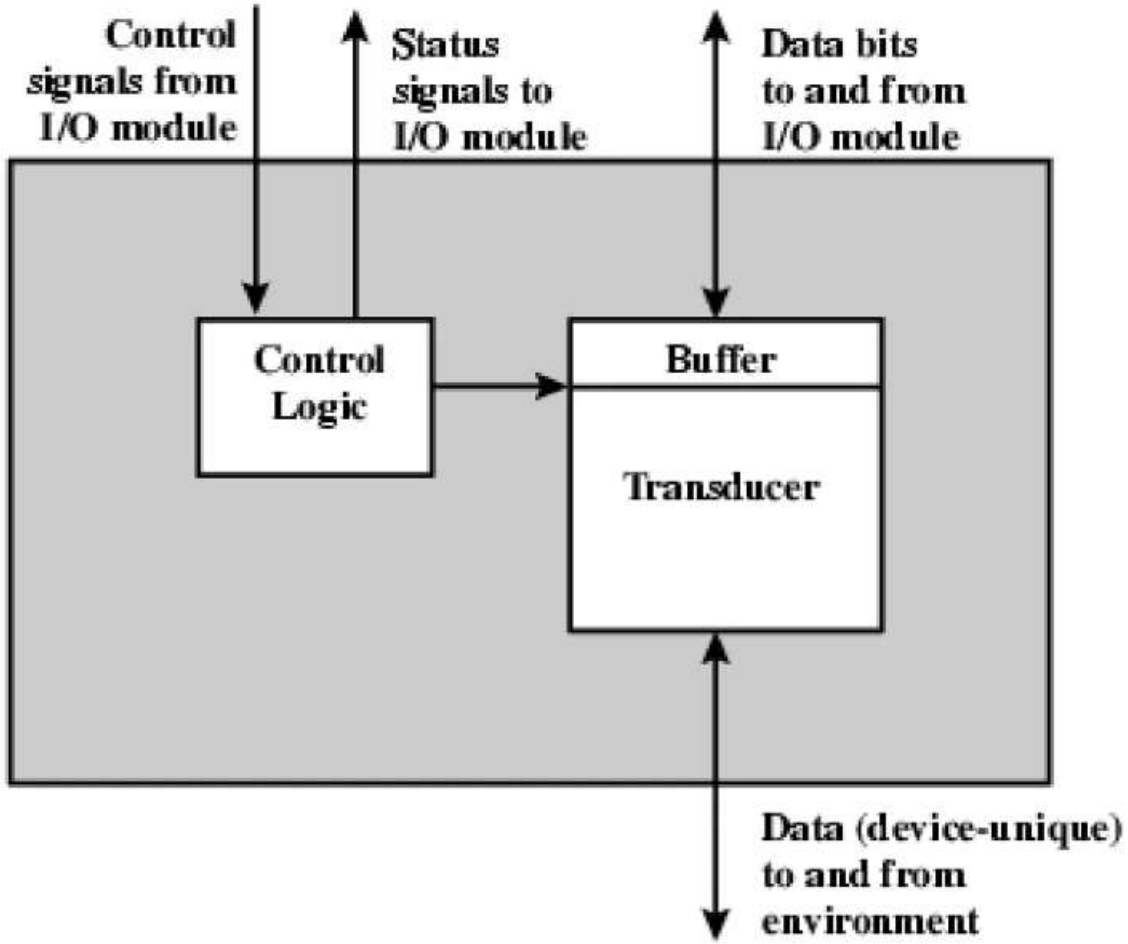
Control logic controls the device operation in response to the direction from I/O module.

Transducer converts data from electrical to other form of energy during output and vice versa.

Buffer temporarily hold data being transferred.

Status signal indicates if the device is ready or not.

Control signal determines the function that the device will perform such as send data to the I/O module or perform some control functions particular to the device



I/O Module Function

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

CONTROL AND TIMING:

- To coordinate the flow of traffic between internal resources and external devices.
- Processor checks the status of attached device through input output module, the module returns the device status.

PROCESSOR COMMUNICATION:

- Command decoding:
  The I/O module accepts commands from the processor typically sent as signals on the control bus.

- Data:
  Data are exchanged between the processor and the I/O module over the data bus.

- Status reporting:
  To check the status of I/O module to synchronize the speed between processor and I/O module.
  eg: BUSY, READY

- Address recognition:
  An I/O module must recognize one unique address for each peripheral it controls.

DEVICE COMMUNICATION:

- This function involves commands status information, and data.

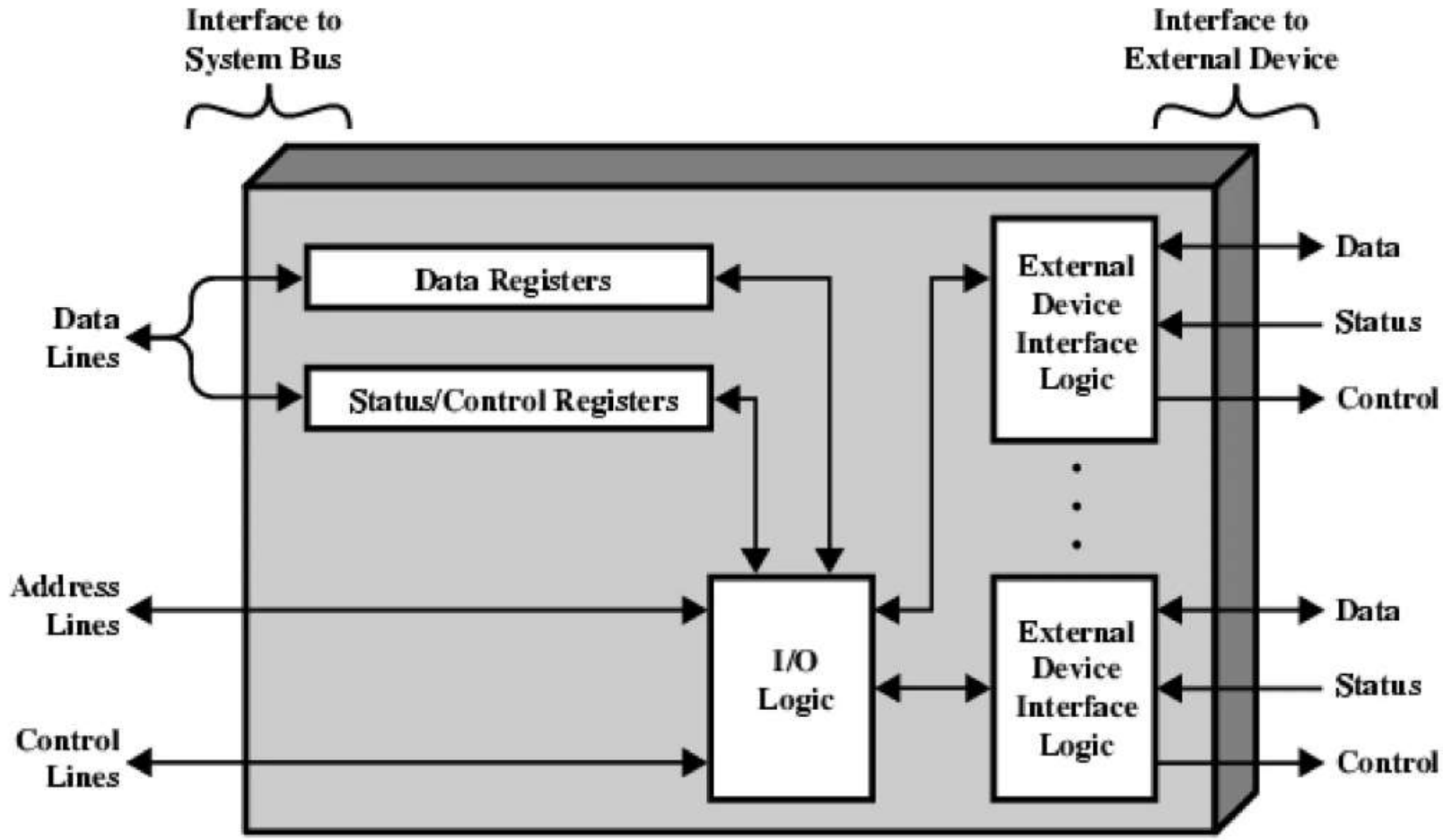
DATA BUFFERING:

- Data coming from main memory are sent to an I/O module in a rapid burst.
- The data are buffered in the I/O module and then sent to the peripheral device at its data rate.
- In opposite direction, data are buffered so as not to tie up the memory in a slow transfer operation.(Data rate synchronization)

ERROR DETECTION:

- It includes mechanical and electrical malfunctions.
- It also includes unintentional change to the bit pattern as it is transmitted from device to I/O module.

I/O Module Diagram



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 Commands

- CPU issues address
  - —Identifies module (& device if >1 per module)
- CPU issues command
  - —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

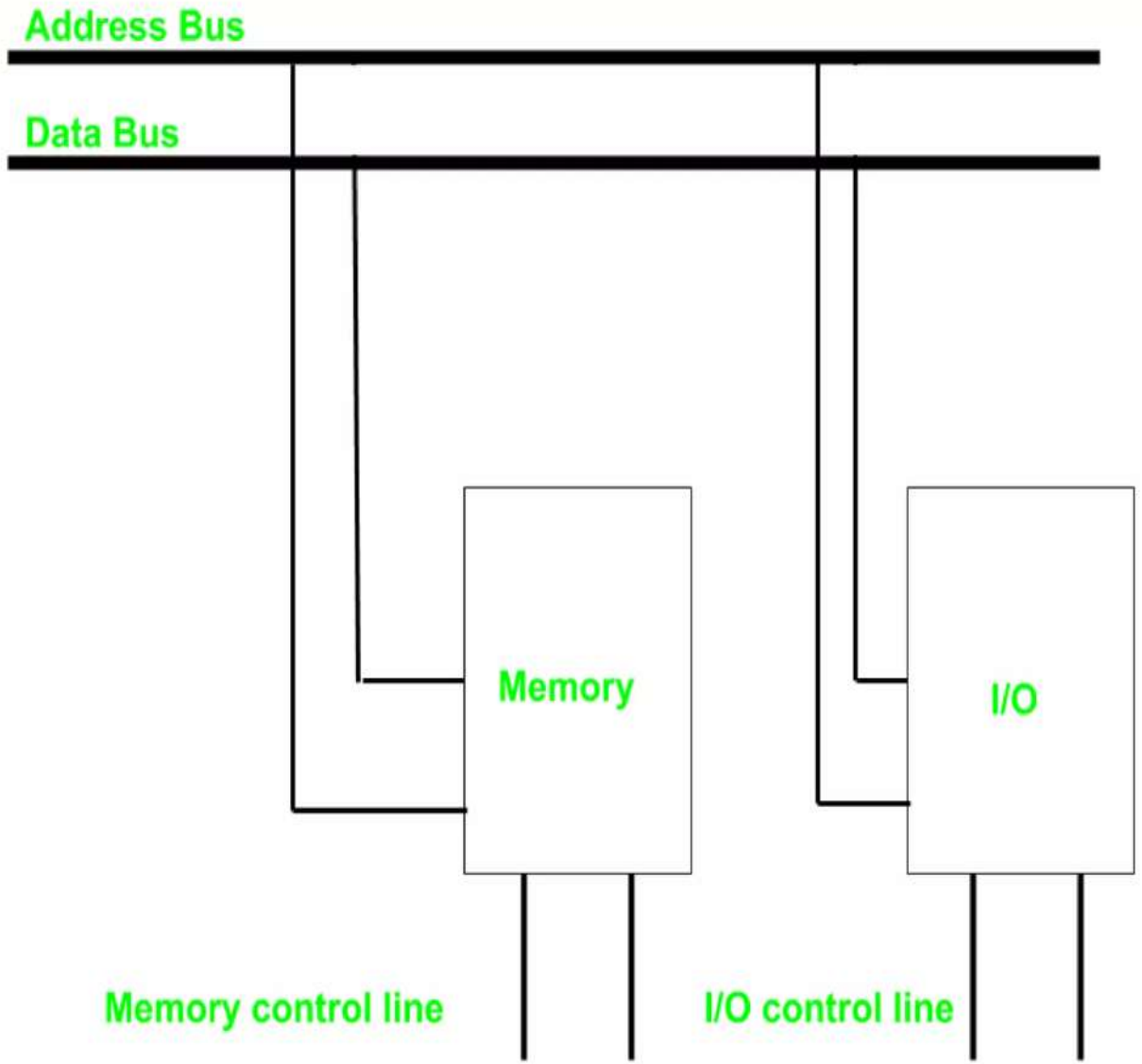
Addressing I/O Devices

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

I/O Mapping

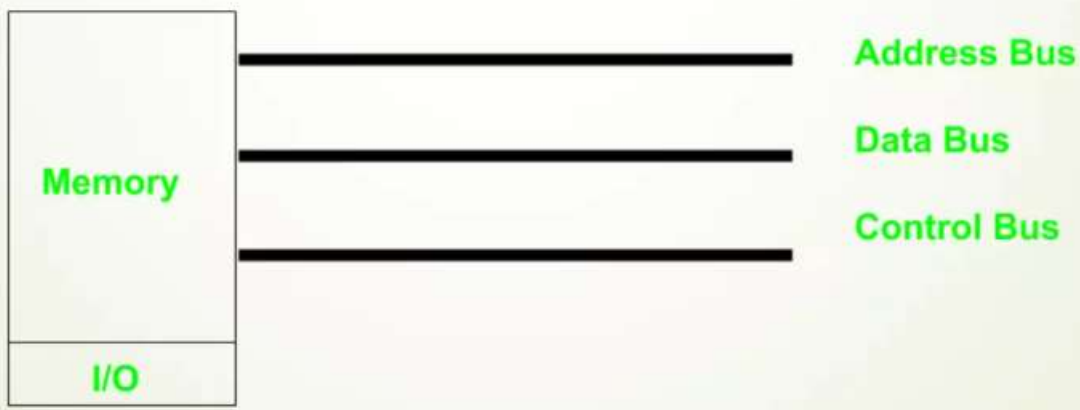
- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set

- Then we have Isolated I/O in which we Have common bus(data and address) for I/O and memory but separate read and write control lines for I/O. So when CPU decode instruction then if data is for I/O then it places the address on the address line and set I/O read or write control line on due to which data transfer occurs between CPU and I/O. As the address space of memory and I/O is isolated and the name is so. The address for I/O here is called ports. Here we have different read-write instruction for both I/O and memory.



Memory Mapped I/O –

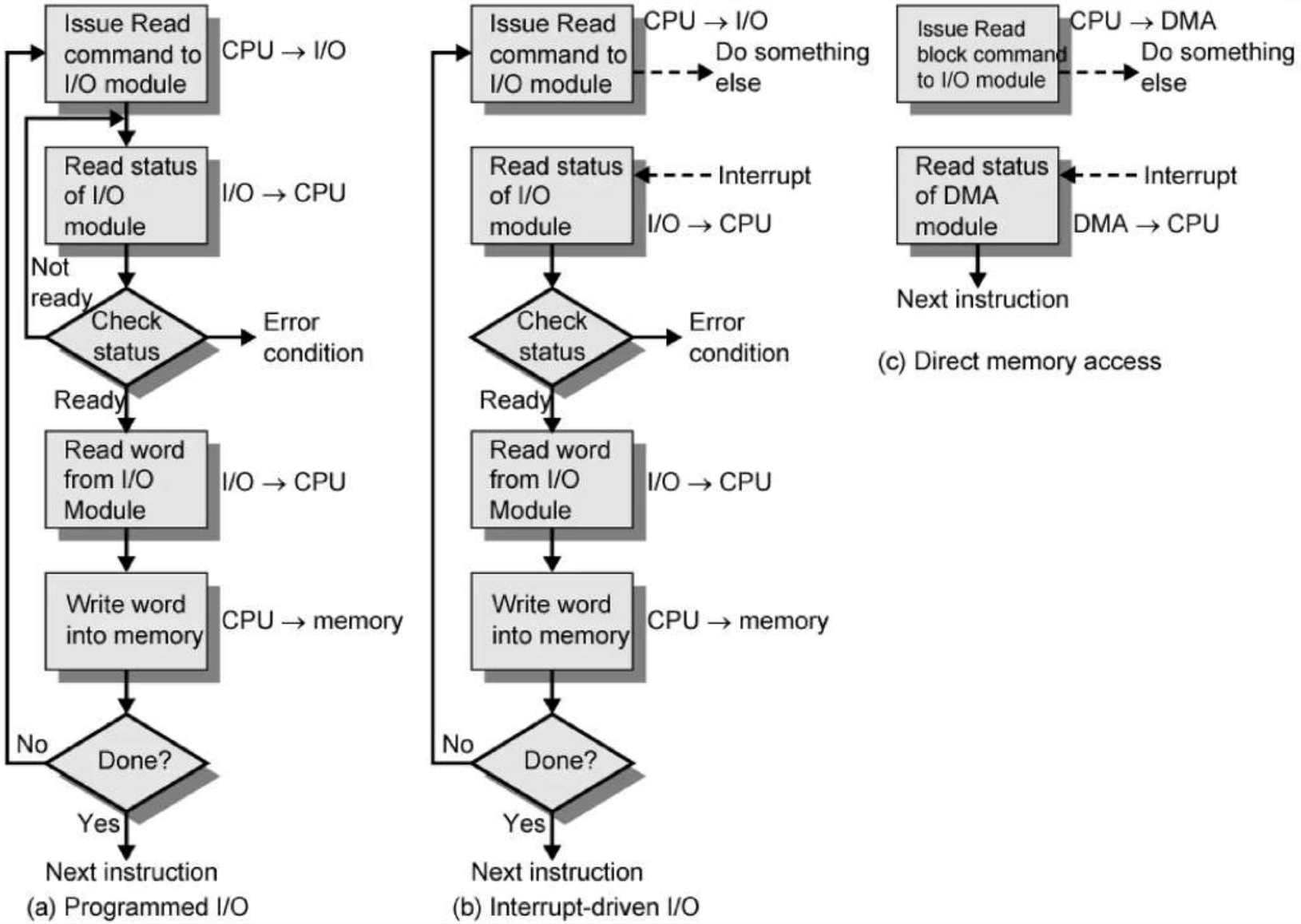
- In this case every bus in common due to which the same set of instructions work for memory and I/O. Hence we manipulate I/O same as memory and both have same address space, due to which addressing capability of memory become less because some part is occupied by the I/O.



Input Output Techniques

- Programmed
- Interrupt driven
- 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

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

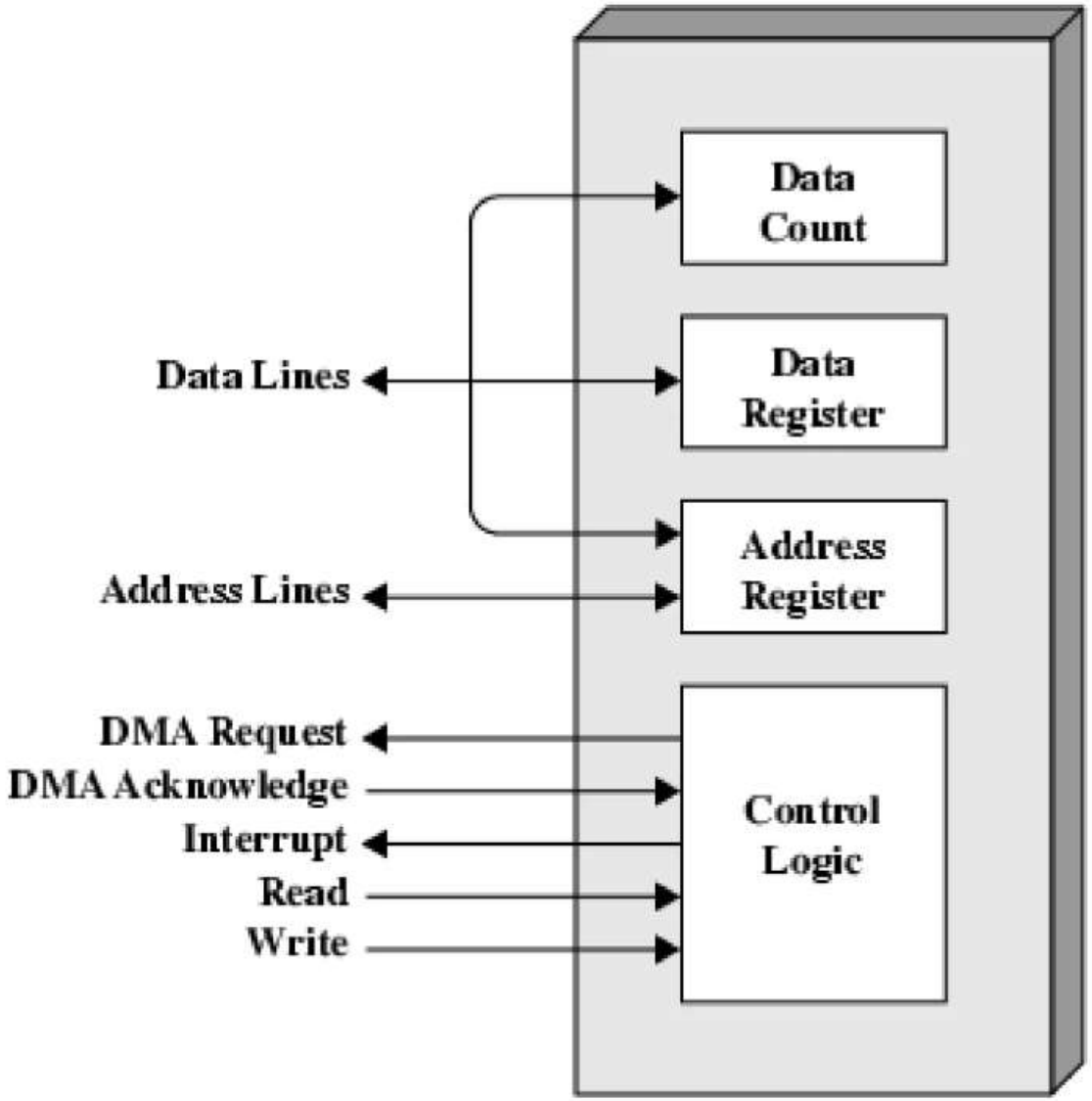
Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - —Transfer rate is limited
  - —CPU is tied up
- 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

1.Burst or Block Transfer DMA

In this scheme the I/O device withdraws the DMA request only after all the data bytes have been transferred.
In this mode the DMA controller acts the master.

2.Cycle steal or Single-Byte transfer DMA

In this mode, only a single byte is transferred at a time.

In this scheme the bytes are divided into several parts and after transferring every part the control of buses is given back to MPU and later stolen back when MPU does not need it.

DMA Transfer

Cycle Stealing

- DMA controller takes over bus for a cycle
- Transfer of one word of data
- 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

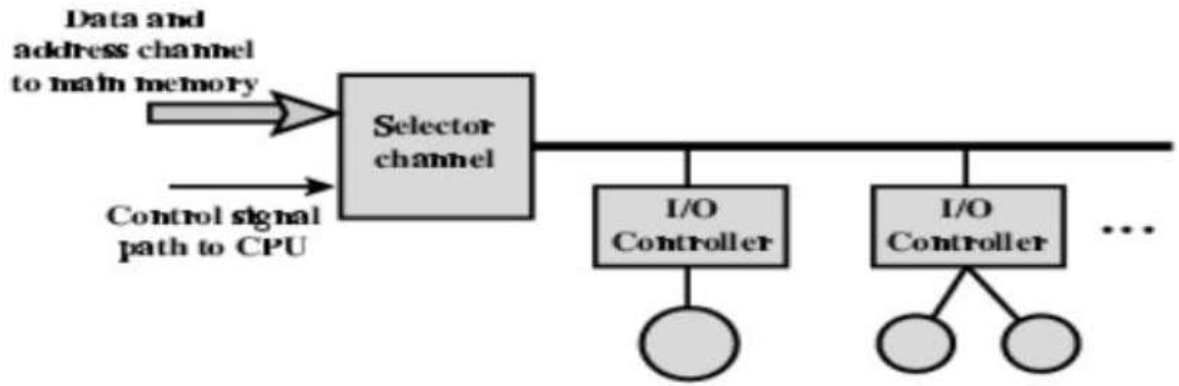
- Advantages of DMA
  - Computer system performance is improved by direct transfer of data between memory and I/O devices, bypassing the CPU.
  - CPU is free to perform operations that do not use system buses.

- Disadvantages of DMA
  - In case of Burst Mode data transfer, the CPU is rendered inactive for relatively long periods of time.

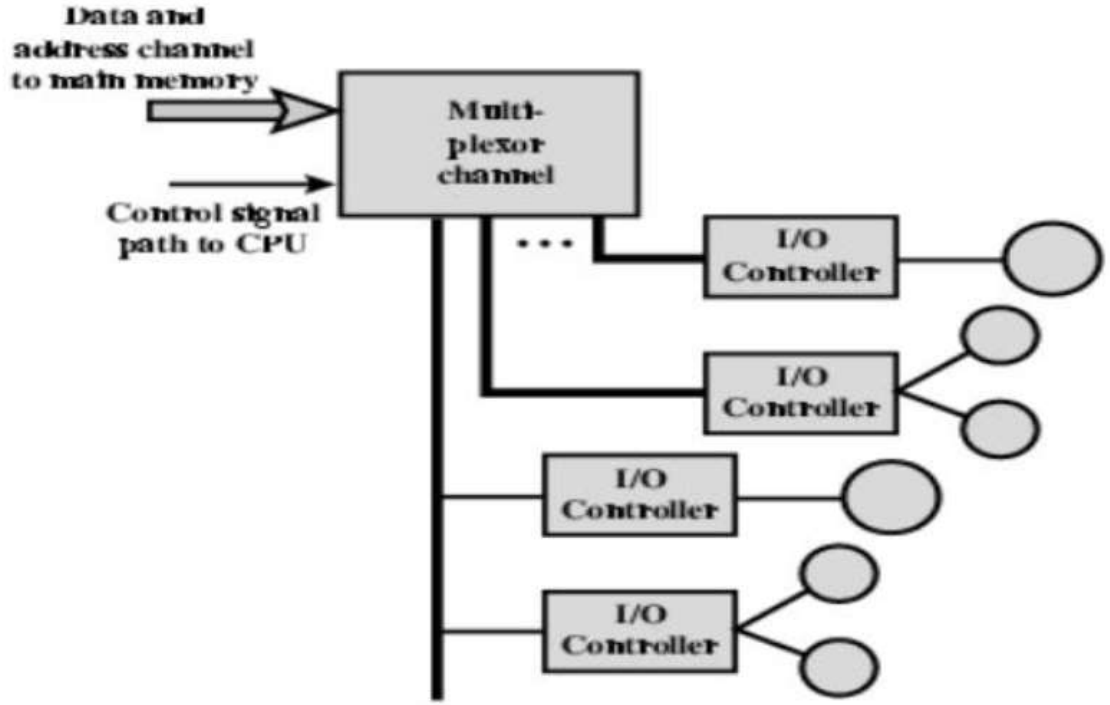
I/O Channels

- I/O devices getting more sophisticated – have a dedicated processor ≈ I/O Channel
  - e.g. 3D graphics cards
- CPU instructs I/O channel to execute an I/O program
- I/O channel handles all transfers
- Improves speed
  - Takes load off CPU
  - Dedicated processor is faster

I/O Channel Architecture



(a) Selector



(b) Multiplexor

Input-Output Processor:

- It is a processor with direct memory access capability that communicates with IO devices.
- IOP is similar to CPU except that it is designed to handle the details of IO operation.
- Unlike DMA which is initialized by CPU, IOP can fetch and execute its own instructions.
- IOP instruction are specially designed to handle IO operation.

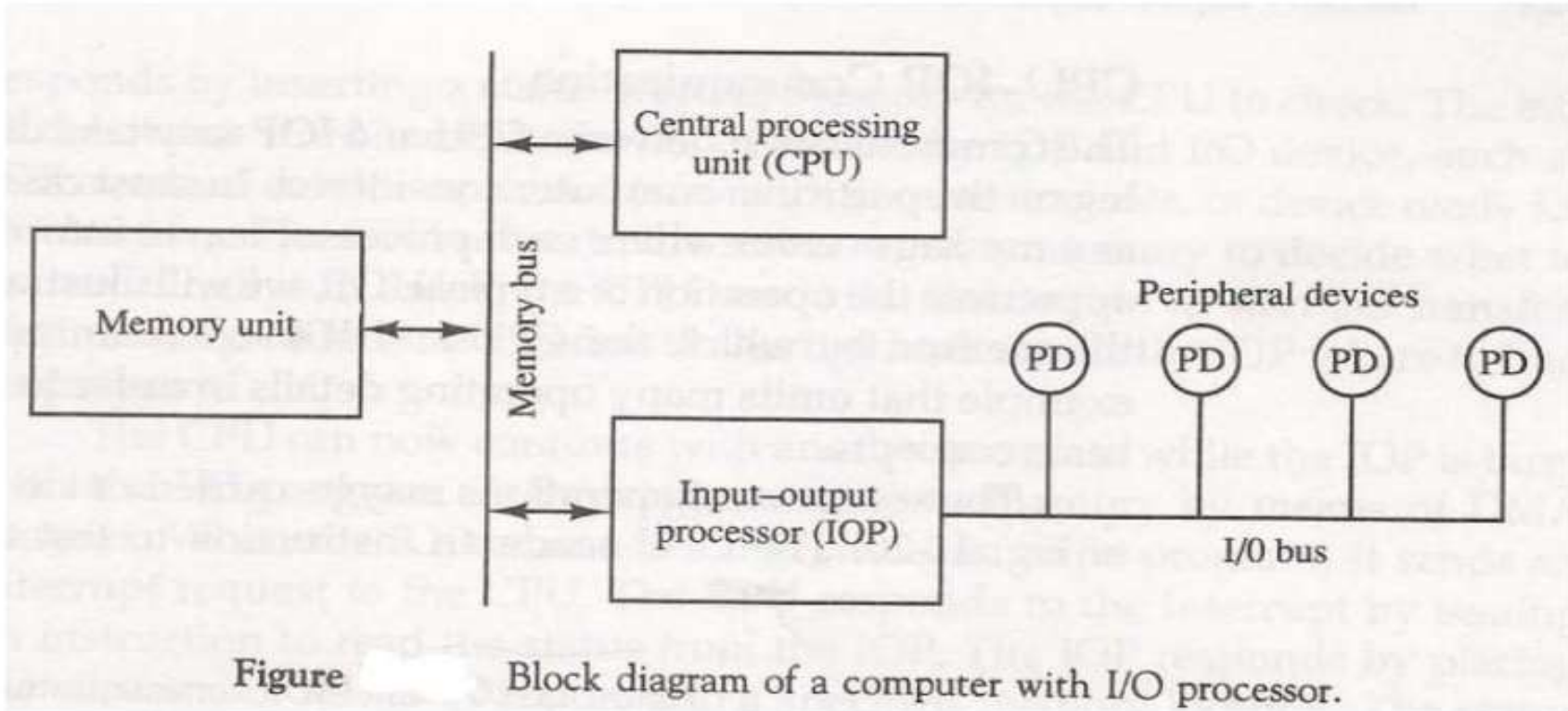


Figure Block diagram of a computer with I/O processor.

- Memory occupies the central position and can communicate with each processor by DMA.
- CPU is responsible for processing data.
- IOP provides the path for transfer of data between various peripheral devices and memory.
- Data formats of peripherals differ from CPU and memory. IOP maintain such problems.
- Data are transfer from IOP to memory by stealing one memory cycle.
- Instructions that are read from memory by IOP are called commands to distinguish them from instructions that are read by the CPU.

Flow chart of CPU-IOP Communication

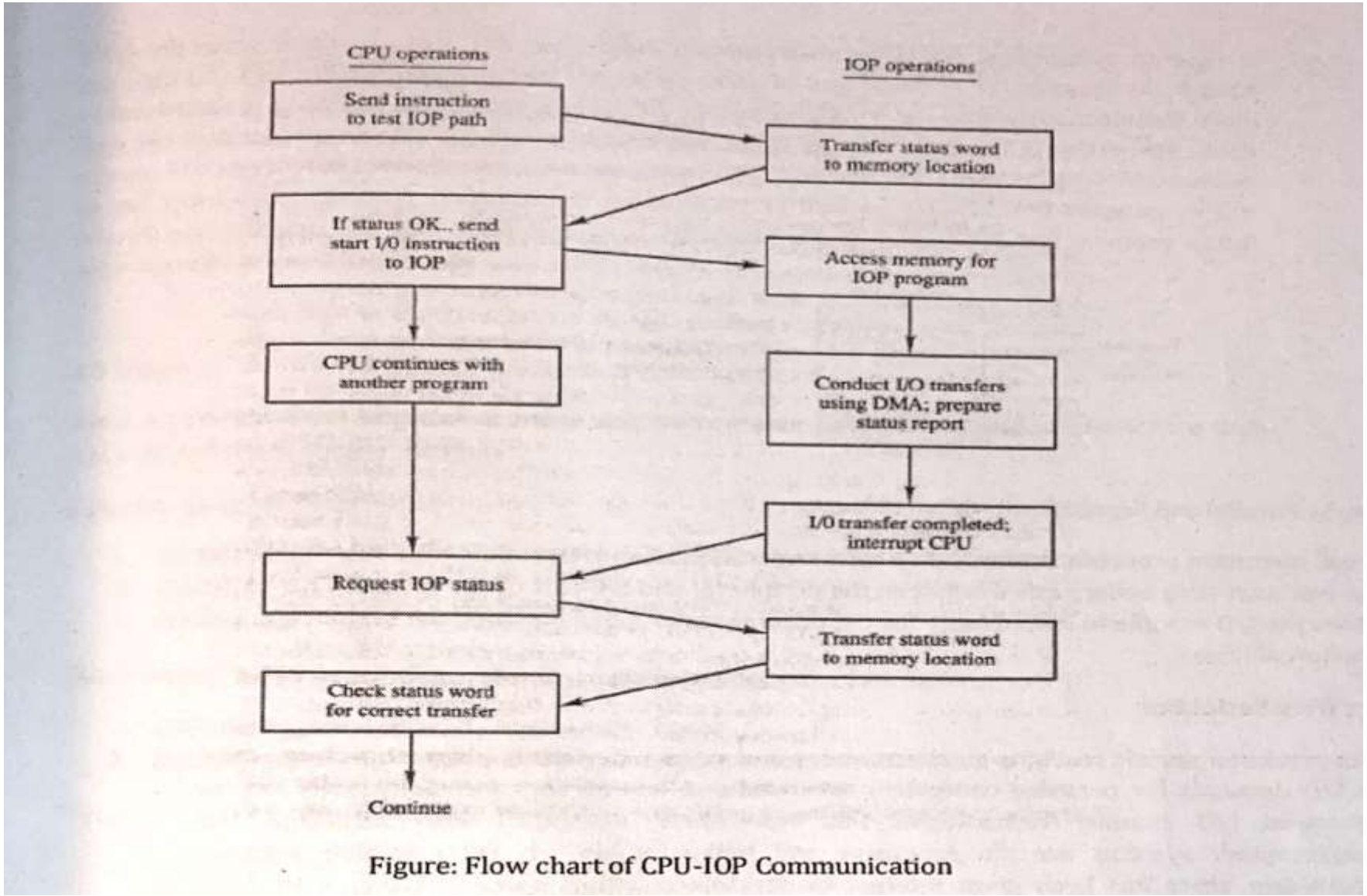


Figure: Flow chart of CPU-IOP Communication

External Interfaces

- Serial or Parallel interface
- I/O module sends a control signal requesting permission to send data, peripheral send acknowledgement.
- I/O module transfer data(serial or parallel)
- Peripheral acknowledges receipt of the data
- Internal buffer store data being passed

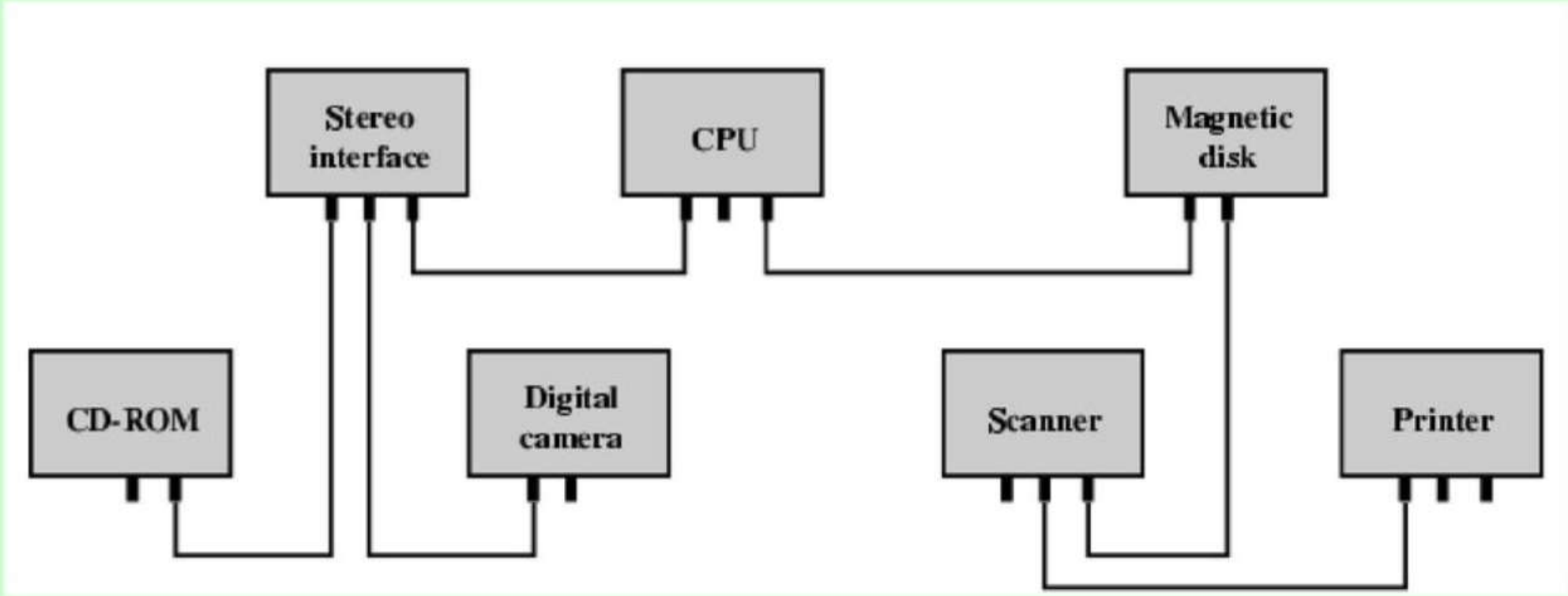
- Connecting devices together
- Bit of wire?
- Dedicated processor/memory/buses?
- E.g. FireWire, InfiniBand

IEEE 1394 FireWire

- High performance serial bus
- Fast
- Low cost
- Easy to implement
- Also being used in digital cameras, VCRs and TV

- Daisy chain
- Up to 63 devices on single port
  - —Really 64 of which one is the interface itself
- Up to 1022 buses can be connected with bridges
- Automatic configuration
- No bus terminators
- May be tree structure

Simple FireWire Configuration

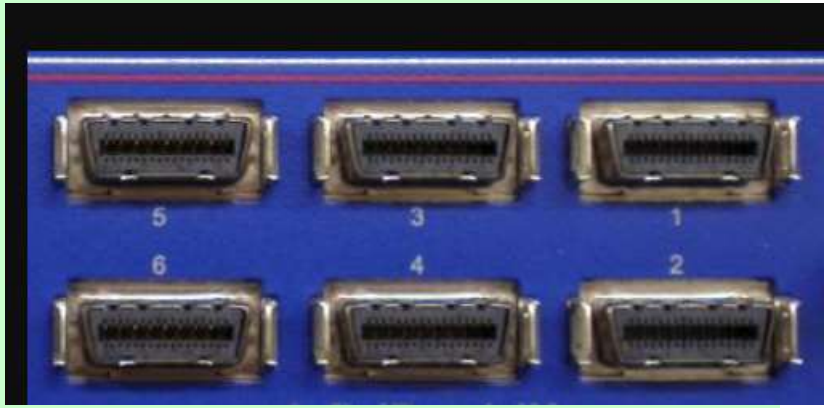


InfiniBand

- I/O specification aimed at high end servers
  - —Merger of Future I/O (Cisco, HP, Compaq, IBM) and Next Generation I/O (Intel)
- Version 1 released early 2001
- Architecture and spec. for data flow between processor and intelligent I/O devices
- Intended to replace PCI in servers
- Increased capacity, expandability, flexibility

InfiniBand Architecture

- Remote storage, networking and connection between servers
- Attach servers, remote storage, network devices to central fabric of switches and links
- Greater server density
- Scalable data centre
- Independent nodes added as required
- I/O distance from server up to
  - 17m using copper
  - 300m multimode fibre optic
  - 10km single mode fibre
- Up to 30Gbps



Multiple Bus Hierarchies

- Why?

1. The more the devices connected, the greater the propagation delay hence less performance
2. Performance can be improved by increasing data rate and using wider databus

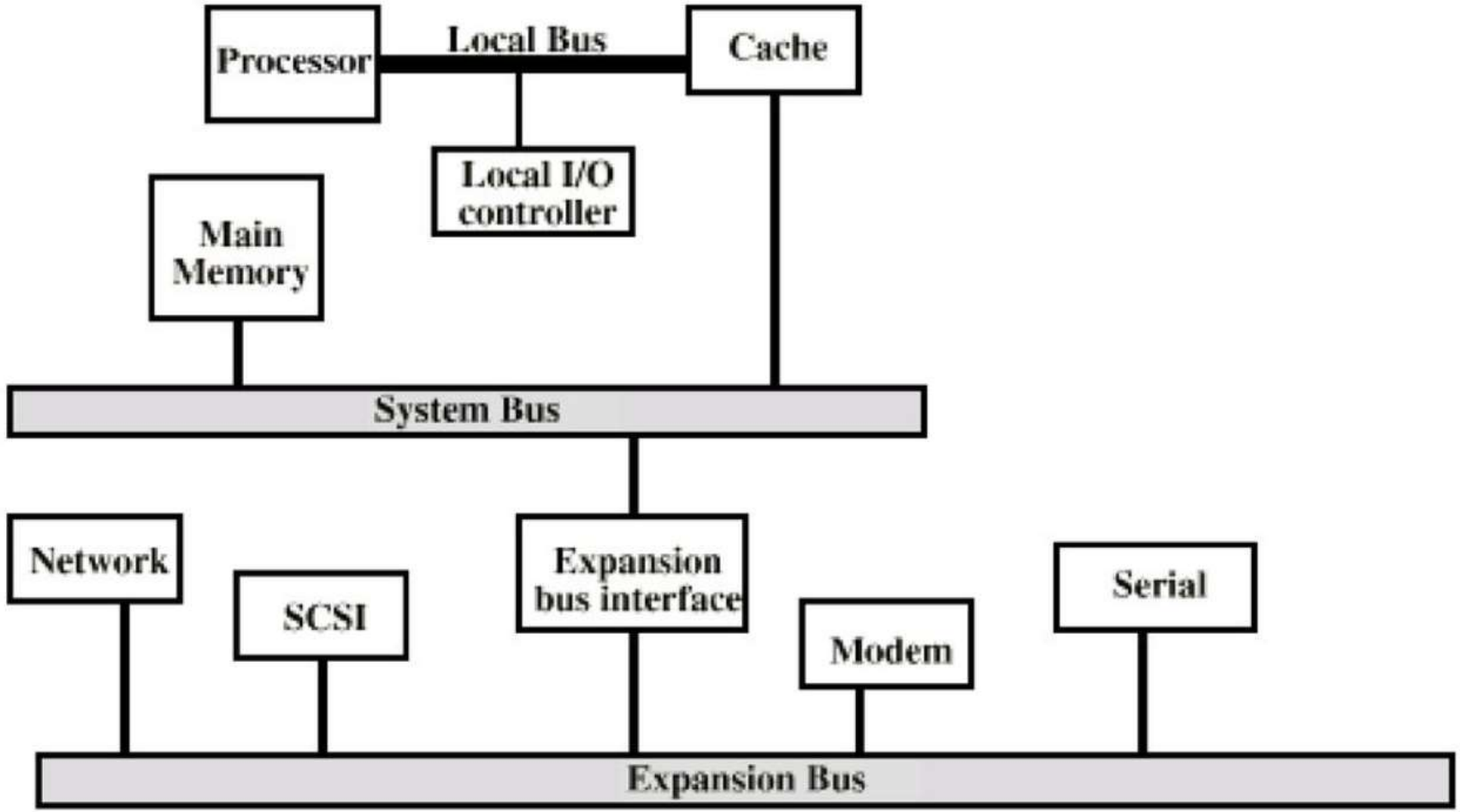
- Two Approaches :

1. Traditional Bus Architecture
2. High Performance Architecture

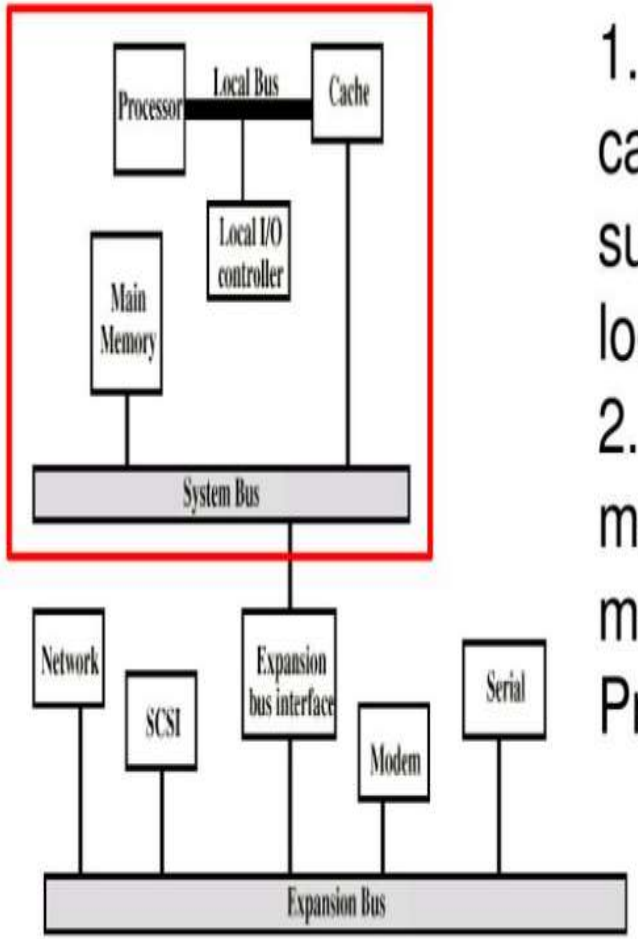
Traditional Bus Architecture

- Isolates processor to memory traffic from I/O traffic.
- Cache Memory act as an interface to system bus
- Expansion bus interfaces it to external devices

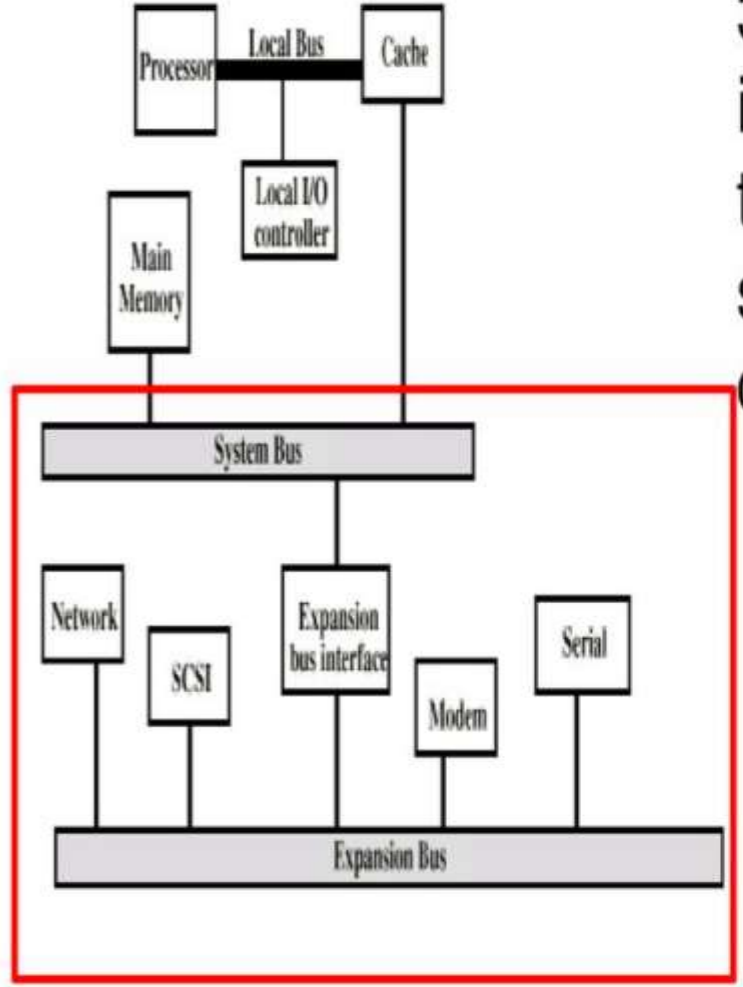
Traditional Bus Architecture



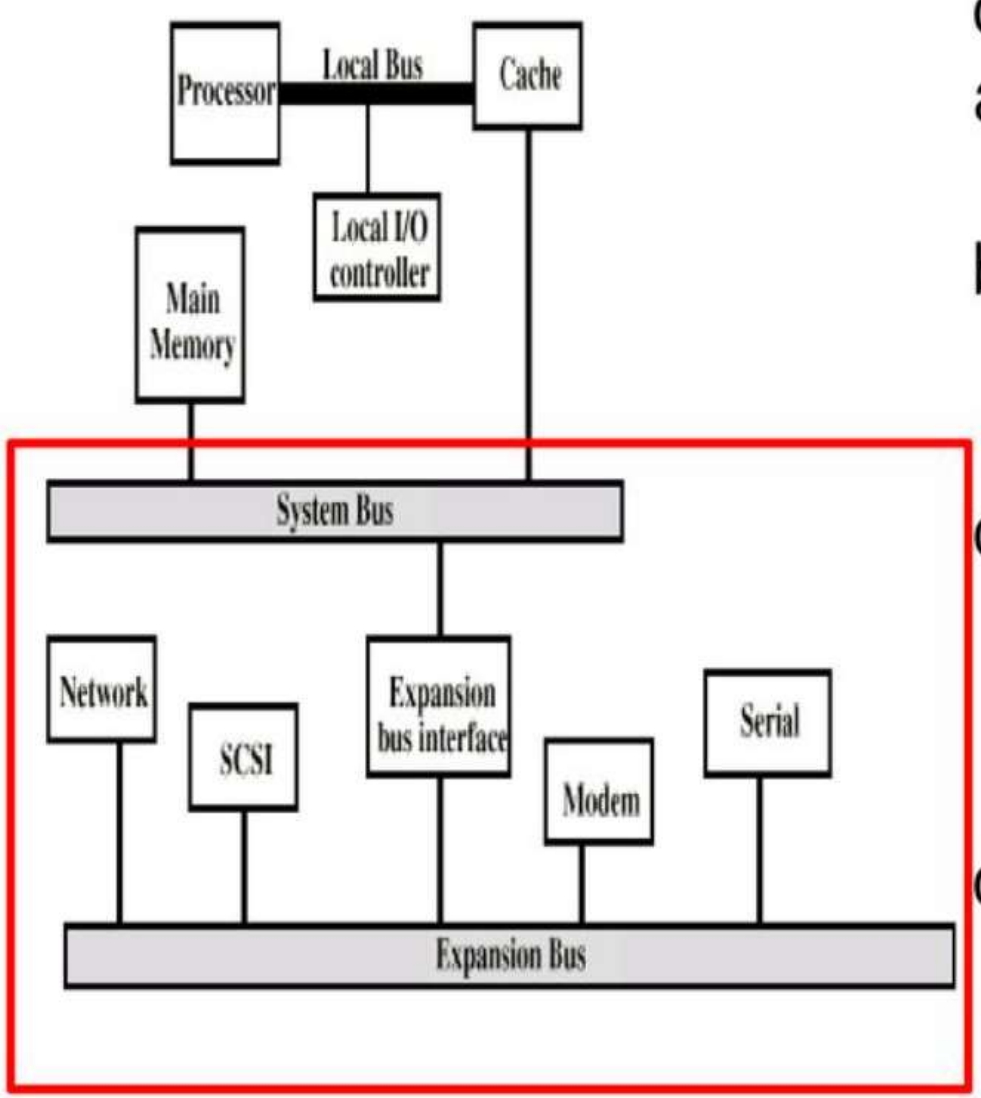
Traditional Bus Architecture



1. Local bus connects cache memory and support one or more local devices
2. Cache is attached to main memory,isolates main memory and Processor



3. Expansion Bus interface buffers data transfers between system bus and I/O controllers



4. Some examples of I/O devices attached:

a) Network : Local Area Network(LAN) (10 Mbps)

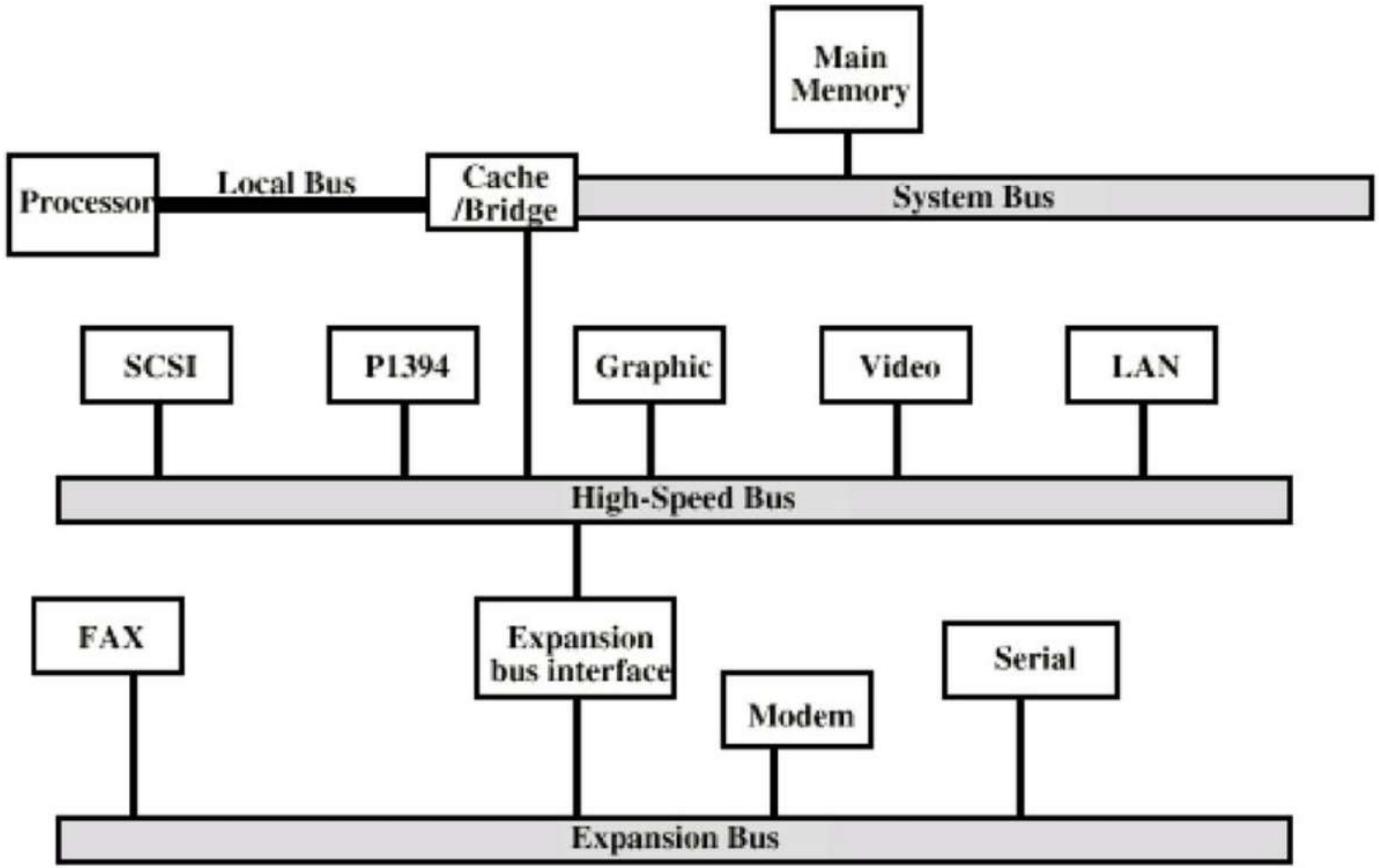
b) Modem : To connect to Wide Area Network(WAN)

c) Small Computer System Interface (SCSI): To support local disk drives and peripherals

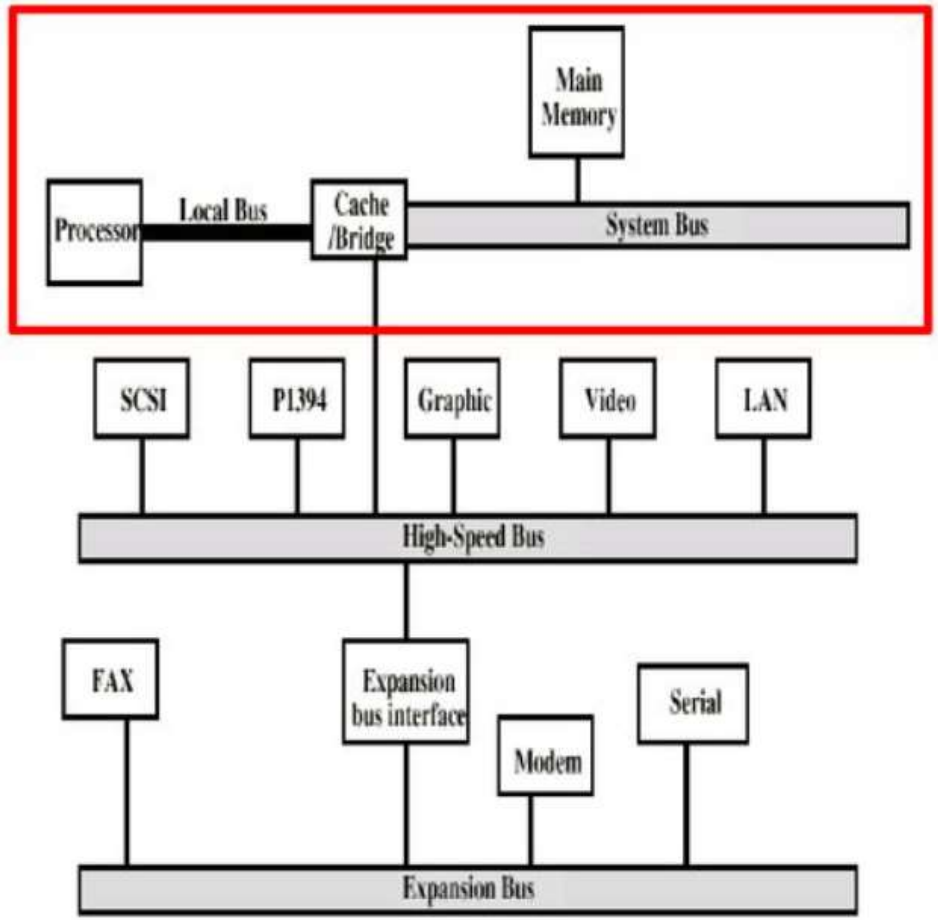
d) Serial Port : for printer and Scanner

SCSI

High Performance Architecture

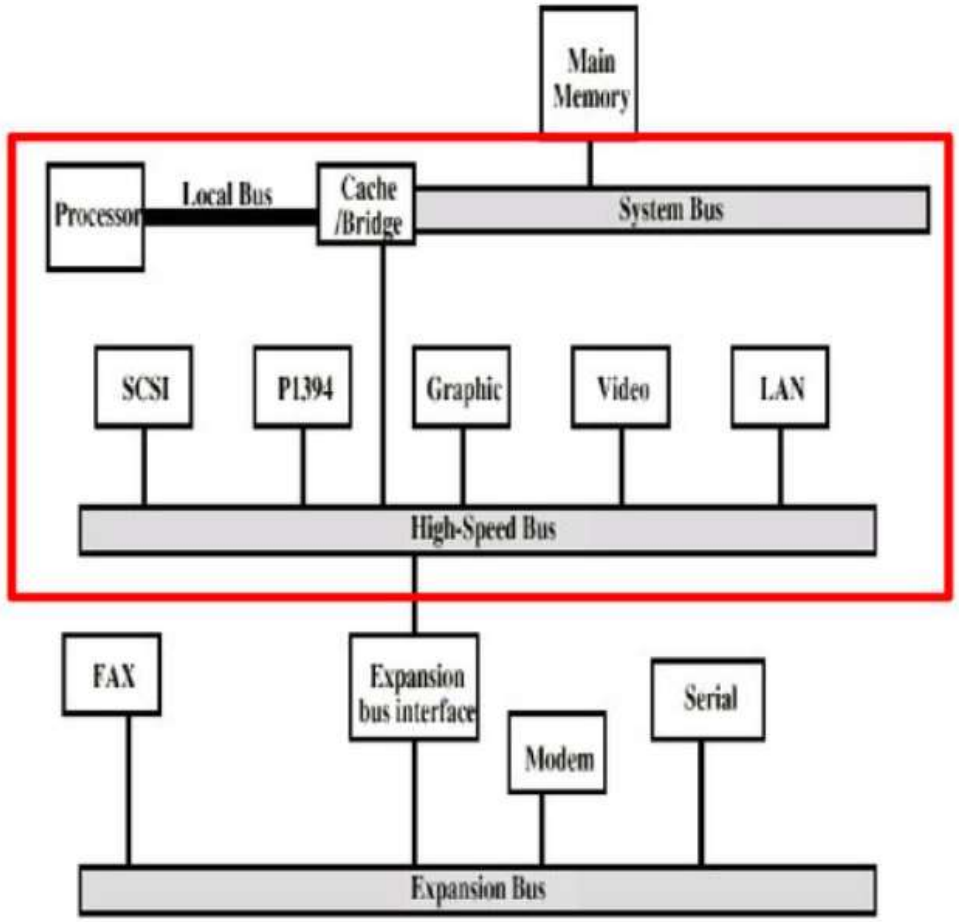


High Performance Architecture



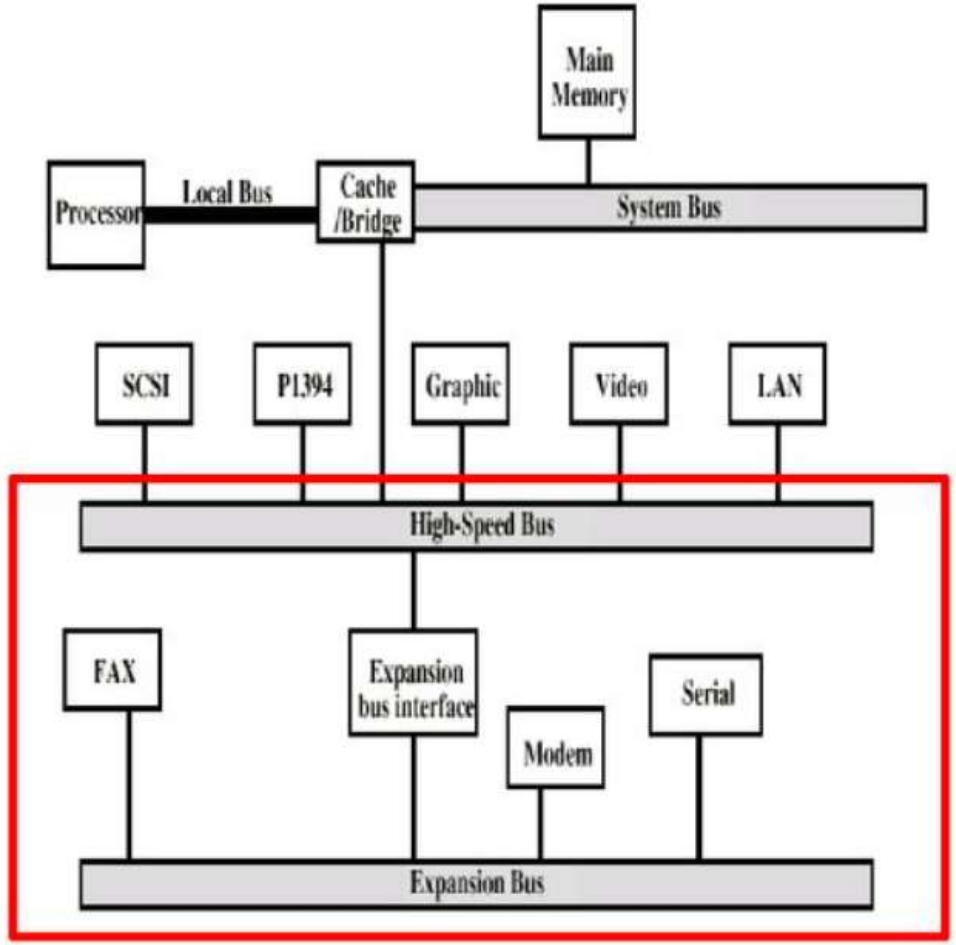
1) Local bus connects processor to cache controller which is connected to main memory via system bus

High Performance Architecture



1) Cache controller is integrated to a bridge (buffering device) that connects high speed bus
2) High Speed Bus connects to
   a) High Speed LANs (100 Mbps)
   b) Video and Graphic Workstation controllers
   c) Interface Controllers to local peripheral buses- SCSI and Firewire

High Performance Architecture



Lower Speed devices are supported with an expansion bus with an interface buffering traffic between the expansion bus and the high speed bus.

Advantages

- High Speed Bus brings high demand devices close to processor but independent.
- Differences in the speed is tolerated
- Changes in processor architecture does not affect high speed bus and vice versa

Elements of Bus Design

- The parameters that classify buses are

1. Bus Types
2. Method of Arbitration
3. Timing
4. Bus Width
5. Data Transfer Type

Bus Types

- Dedicated and Multiplexed
- **Dedicated**
  - Bus line is permanently assigned to a function or a subset of computer components
  - It uses multiple buses
  - Adv: High throughput and less bus contention
  - Disadv: increased size and cost
- **Multiplexed:**
  - Same bus is used for different functions
  - E.g. Address and data may be transmitted over same set of lines
  - Adv: use of few lines saves space and cost
  - Disadv: complex circuitry is needed and less performance

Method of Arbitration

- Centralized and Distributed
- **Centralized :**
  - A single hardware called bus controller allocates time on bus
- **Distributed :**
  - Each module contains access control logic and modules act together to share the bus

Timing

- Synchronous and Asynchronous
- **Synchronous:**
  - Occurrence of events are controlled by clock
  - All events start at the beginning of clock cycle
  - **Adv:** Simple to implement and test
  - **Disadv:** less flexible – cannot take advantage of device performance

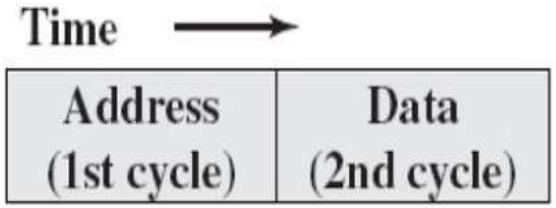
- **Asynchronous:**
  - Occurrence of one event on bus follows the occurrence of previous event
  - **Adv:** fast and slow device can share the bus
  - **Disadv:** Difficult to implement

Bus Width

- Address Bus and Data Bus
- **Address Bus:**
  - Wider address bus → greater range of locations
- **Data Bus**
  - Wider data bus → greater number of bits per unit time

Data Transfer Type

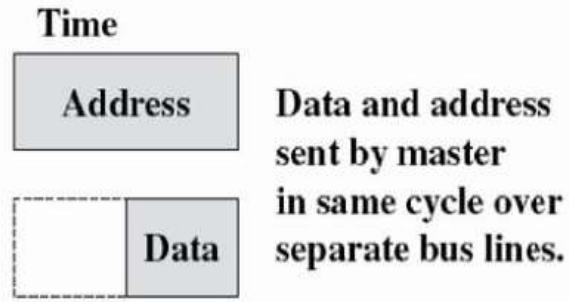
Multiplexed Address/Data Bus



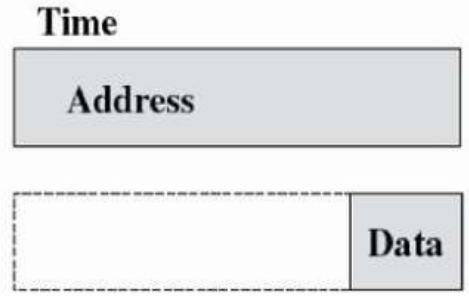
Write (multiplexed) operation

Read (multiplexed) operation

Dedicated Address/Data Bus



Write (non-multiplexed) operation



Read (non-multiplexed) operation

Read Modify Write

Read-modify-write operation

- Read followed by immediate write to the same address
- Used to protect shared memory resources

Read After Write

Read-after-write operation

- Write is immediately followed by Read
- It is for checking purpose

Block Data Transfer

Block data transfer

- One address cycle is followed by n data cycles