Chapter 7: Input/output organization

An important key element of any computer system is I/O (Input Output) module. Different modules interface with system bus or central switch, or different peripheral devices connect to processor through I/O module. So an I/O module is not just a simple mechanical connector that connects a device to the computer but rather contains logic for performing a communication functions between the peripheral and bus.

Some of the reason why we cannot directly connect peripheral devices to the system bus:

1. There are a wide variety of peripherals with different methods of operations, so it would be practically impossible to define methods for every peripheral device within a single processor.
2. The data transfer rate some of the peripheral are slower and some faster than that of memory or processor, so we cannot directly connect to system bus due to this speed mismatch.
3. Peripherals often use different data formats and word length than the computer they are attached to.

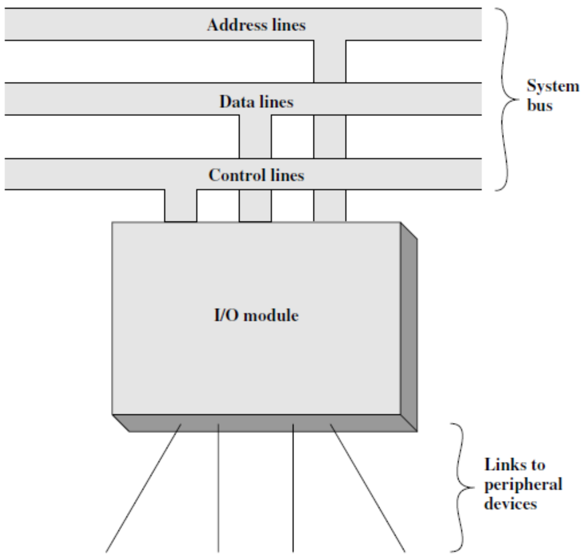


Figure: generic model of I/O devices

4.1 External devices

External devices are those devices that provide a way of communicating medium between external environment and the computer. An external device is attached to the computer through a link to an I/O module. The link is used to exchange control, status, and data between I/O module and external devices. We can broadly classify external devices into three categories as:

Human readable: those used for communicating with computer user, example: video display terminal and printers.

Machine readable: those used for communicating with equipment. Example magnetic tapes, magnetic disks.

Communications: those used for communicating with remote devices, i.e. devices that allows to exchange of data with remote devices.

In general terms the nature of external devices is indicated in the figure. The interface module to the I/O module is in the form of control, data and status signal.

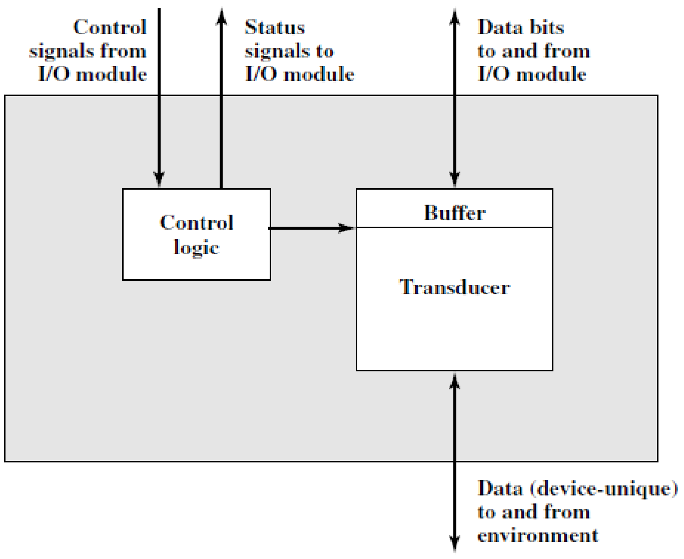

Control signals from I/O module
Status signals to I/O module
Data bits to and from I/O module
Control logic
Buffer
Transducer
Data (device-unique) to and from environment

Figure: Block diagram of external devices

Control signal determines the function that the device will perform such as send data to the I/O module, accept data from I/O module, report status or perform some control functions particular to the device.

Data are transferred in the form of bits to be sent or received from I/O module.

Status signals indicate the state of the devices like device ready or not.

Control logic associated with the device controls the device's operation in response to the direction from I/O module. The transducer converts data from electrical to other forms of energy during output and from other form to electrical during input. A buffer is associated with the transducer to temporarily hold data being transferred.

Example of external devices

The most common means of computer/user interaction is a keyboard/monitor arrangement. The user provides input through the keyboard. This input is then transmitted to the computer and may also be displayed on the monitor. The basic unit of exchange is the character. Associated with each character is a code, typically 7 or 8 bits in length. The most commonly used text code is the International Reference Alphabet (IRA) or American Standard Code for Information Interchange (ASCII). Each character in this code is represented by a unique 7-bit binary code; thus, 128 different characters can be represented. Characters are of two types: *printable* and *control*. Printable characters are the alphabetic, numeric, and special characters that can be printed on paper or displayed on a screen. Some of the control characters have to do with controlling the printing or displaying of characters. For keyboard input, when the user depresses a key, this 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 then transmitted to the I/O module in the computer. At the computer, the text can be stored in the same IRA code. On output, IRA code characters are transmitted to an external device from the I/O module. The transducer at the device interprets this code and sends the required electronic signals to the output device either to display the indicated character or perform the requested control function.

Disk Drive

A disk drive contains electronics for exchanging data, control, and status signals with an I/O module plus the electronics for controlling the disk read/write mechanism. In a fixed-head disk, the transducer is capable of converting between the magnetic patterns on the moving disk surface and bits in the device's buffer.

4.2 I/O Modules

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

- Control and timing
- Processor communication
- Device communication
- Data buffering
- Error detection

The internal resources, such as main memory and the system bus, must be shared among a number of activities, including data I/O. Thus, the I/O function includes a *control and timing* requirement, to coordinate the flow of traffic between internal resources and external devices.

Processor communication involves the communication between the processor and I/O module. It involves the following:

Command decoding: The I/O module accepts the command from processor sent as signals on control bus and decodes it.

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

Status reporting: It is important to know the status of the I/O module because of the speed mismatch between the processor and I/O module. E.g. BUSY and READY status signals.

Address recognition: I/O module must recognize the unique address of the I/O peripherals that it controls.

The I/O module must be able to perform *device communication*. This communication involves commands, status information, and data. The transfer rate into and out of main memory or the processor is quite high so we need a *data buffering* to match the different data rates of the peripheral devices. An I/O module is often responsible for error detection and subsequently reporting errors to the processor, example of errors includes mechanical and electrical malfunctions reported by the device like, paper jam, bad disk track.

4.2.1 I/O Module structure

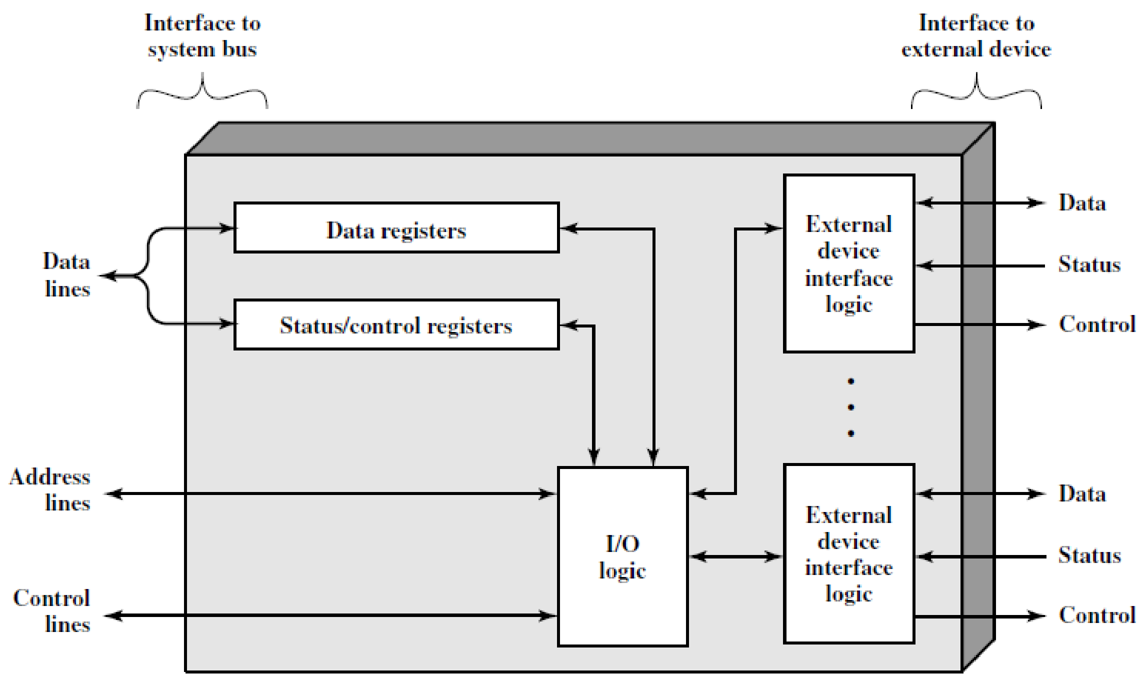


Figure: Block diagram of an I/O module

I/O modules vary considerably in complexity and the number of external devices they control. The module is connected to computer through a *system bus*. Data transfer to and from the module are buffered in one or more *data registers*. There may be one or more *status registers* that provide current status information. A status register may also function as a *control register* to accept detailed control information from processor. The logic within the module interacts with the processor via a set of control lines to issue commands to I/O module.

Each I/O module has a unique address or if it controls more than one external device a unique set of address. I/O module contains specific logic to interface with each device that it controls. I/O module may hide details of timing, formats and mechanics of external device so that processor can function in terms of simple read and write commands.

4.3 I/O techniques

Three techniques are possible for I/O operations:

1. Programmed I/O
2. Interrupt Driven I/O
3. Direct Memory access(DMA)

1. Programmed I/O

Programmed I/O is result of I/O instructions written in computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer data to and from CPU to memory. Transferring the data under program control requires constant monitoring of the peripheral by CPU, but in programmed I/O technique once the data transfer is initiated, the CPU is required to monitor the interface to see when again a transfer can be made.

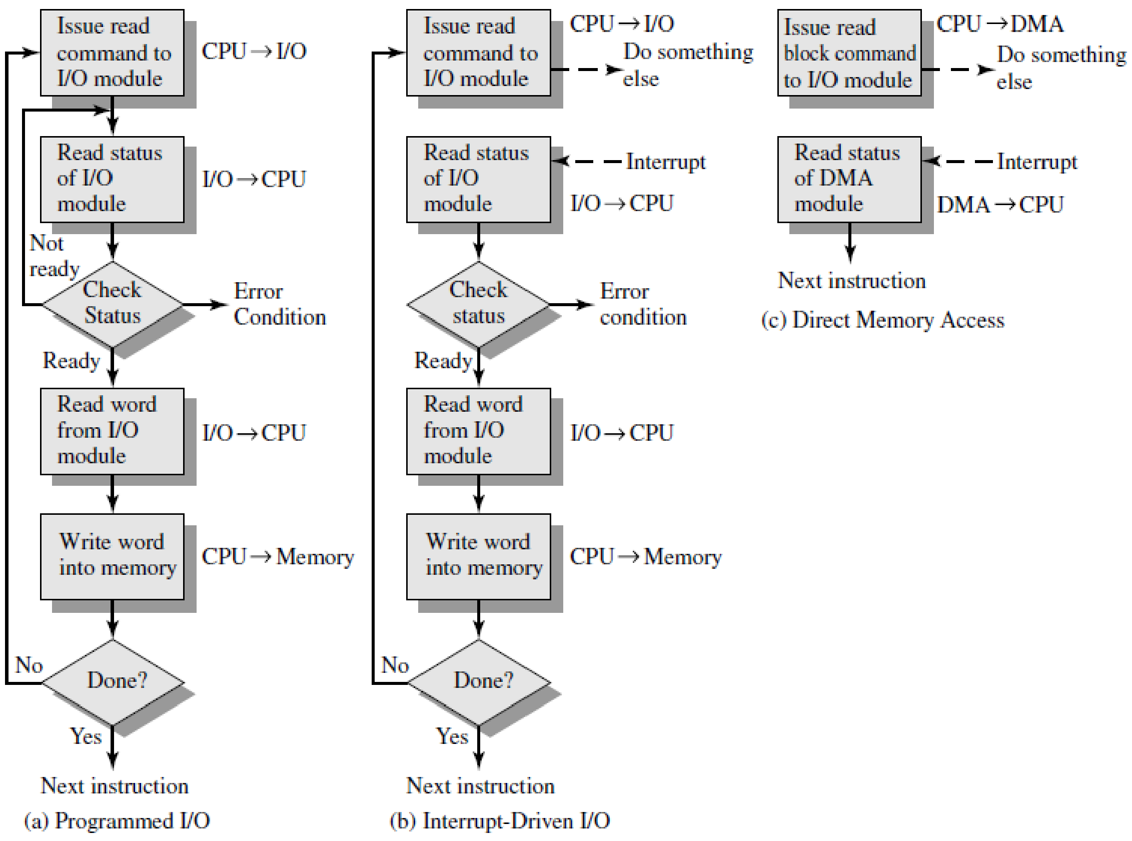


Figure: Flow chart of three techniques

Steps followed in the programmed I/O techniques

- CPU issues the read command to I/O module.
- Read status of the I/O module.
- Check the status of I/O module. If the module is not ready then wait some time and again check for the status, if error condition occurs then run error handling subroutine, if the module is ready then read word from I/O module.
- Then if needed write word into memory.
- Check if all tasks are completed if yes then go to next instruction else start from step 2.

Here we have to analyze the programmed I/O techniques from point of view of I/O commands issued by the processor to the module and then from the point of view of the instruction executed by the processor.

To execute an I/O related instruction the processor issues an address, specifying the particular I/O module and external device, and an I/O command. 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 (byte or word) from the data bus and subsequently transmit that data item to the peripheral.

There will be many I/O devices connected through I/O module to the system, each device is given a unique identifier or address. Thus each I/O module must interpret the address line. When the processor, main memory and I/O share a common bus two modes of addressing are possible:

Memory mapped I/O:

Processor treats the status and data register of I/O module as memory locations and use same machine instruction to access both memory and I/O devices. There is single address space for memory location and I/O devices.

Isolated I/O:

Address space of I/O module is isolated from the memory address space. (The bus may be equipped with memory read and memory write plus input and output command lines. Now the command line specifies whether the address refers to a memory location or I/O devices). Separate instructions in the instruction set are used to perform I/O.

2. Interrupt Driven I/O

The problem with programmed I/O technique is that the processor has to wait a long time for I/O module to be ready for either reception or transmission of data. The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded. And alternative way could be:

- Issue an I/O command to the module by the CPU.
- CPU continues with its other task while the module performs its task.
- Module signals the CPU when I/O operations is finished (i.e. generate interrupt) when it is ready to exchange data.
- CPU responds to the interrupt by executing an interrupt service routine and then continues on with it's with its primary task after interrupt service routine completion.

Advantage of this technique is that CPU involvement is less than programmed I/O technique. CPU recognizes and responds to interrupt at the end of an instruction cycle.

3. Direct memory access (DMA)

Both the programmed and interrupt driven I/O require the continued involvement of the CPU in ongoing I/O operations. Direct Memory Access takes the CPU out of the task except for initialization of the operations. Using this techniques large amount of data can be transferred between memory and peripheral without severely impacting CPU performance.

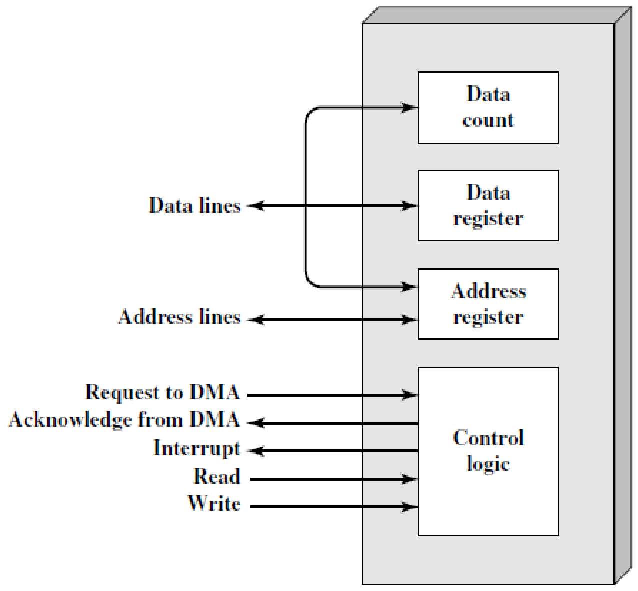


Figure: DMA block diagram

When the processor whishes to read or write a block of data, it issues a command to the DMA module by sending DMA module the following information:

- Whether a read or write is requested, using the read or write control line between the processor and DMA module.
- The address of the I/O device involved communicated through data lines.
- The starting location in memory to read from or write to communicated on the data lines and stored by the DMA module in its register.
- The number of words to be read or written again communicated via the data lines and stored in the data count register.

The processor then continues with other work. The DMA module transfer the entire block of data one word at a time directly to or from memory, without going through the processor. When the transfer is complete the DMA module sends an interrupt signal to the processor thus the processor is involved only at the beginning and end of the transfer.

DMA module must use the bus only when the processor does not need it or it must force the processor to suspend operation temporarily. The later technique is more commonly referred to as *cycle stealing* because the DMA module in effect steals a bus cycle.

4.4 I/O channels and processor

With the evolution of computer we have seen different techniques used in the computer to enhance the performance. Considering the I/O part we can see the following evolutionary steps:

1. The CPU directly controls peripheral devices, earlier stages of computer system.
2. A controller or I/O module is added so that CPU uses programmed I/O without interrupts.
3. The same configuration as in step 2 is used, but now interrupts are employed. So that 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. The CPU directs the I/O processor to execute an I/O program in memory. The I/O processor fetches and executes these instructions without CPU intervention. This allows the CPU to specify a sequence of I/O activities and to be interrupted only when the entire sequence has been performed. For this method I/O module is referred to as an I/O channel.
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. A common use for such architecture has been to control communication with interactive terminals. The I/O processor takes care of most of the tasks involved in controlling the terminals.

As we proceeds along this evolutionary path, more and more of the I/O function is performed without CPU involvement. The CPU is increasingly relieved of I/O-related tasks, improving performance. With the last two steps (5–6), a major change occurs with the introduction of the concept of an I/O module capable of executing a program. For step 5, the I/O module is often referred to as an I/O channel. For step 6, the term I/O processor is often used.

Characteristics of I/O Channels

An I/O channel has the ability to execute I/O instructions, which gives it complete control over I/O operations. In a computer system with such devices, the CPU does not execute I/O instructions. Such instructions are stored in main memory to be executed by a special-purpose processor in the I/O channel itself. Thus, the CPU initiates an I/O transfer by instructing the I/O channel to execute a program in memory. The program will specify the device or devices, the area or areas of memory for storage, priority, and actions to be taken for certain error conditions. The I/O channel follows these instructions and controls the data transfer.

Two types of I/O channels are common, as illustrated in Figure below. A selector channel controls multiple high-speed devices and, at any one time, is dedicated to the transfer of data with one of those devices. Thus, the I/O channel selects one device and effects the data transfer. Each device, or a small set of devices, is handled by a controller, or I/O module, that is much like the I/O modules we have been discussing. Thus, the I/O channel serves in place of the CPU in controlling these I/O controllers. A multiplexor channel can handle I/O with multiple devices at the same time. For low-speed devices, a byte multiplexor accepts or transmits characters as fast as possible to multiple devices.

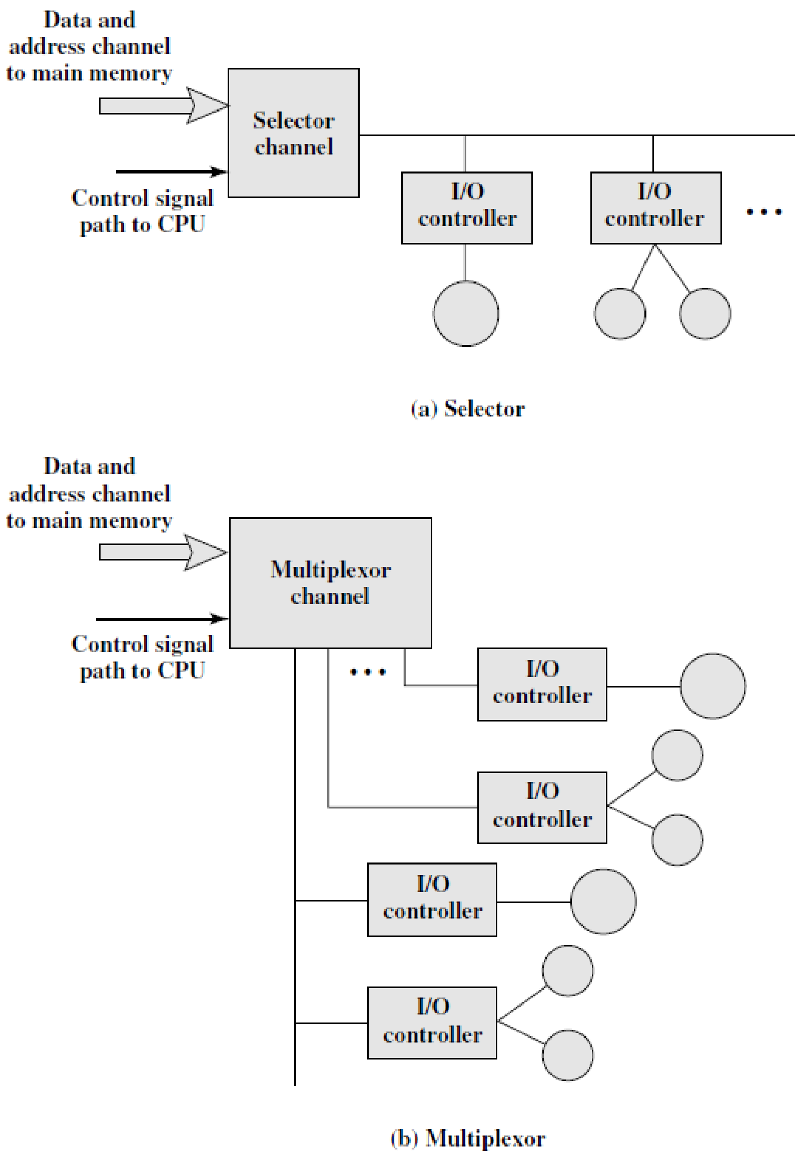


Figure: I/O channels architecture

I/O processor

It is an specialized processor which contains a local memory of its own. It not only loads and stores into memory but also can execute instructions which are among a set of I/O operations. The IOP (Input Output Processor) interfaces to the system and devices. In an IOP based system, I/O devices can directly access the memory without intervention by the processor. The data formats of peripheral devices differ from memory and CPU data formats. The IOP must structure the data words from many different sources to the format of the memory for transfer. Data are gathered in the IOP at device rate and bit capacity while CPU is executing its program. After data are assembled into the memory word they are transferred from IOP directly into the memory by *stealing* one memory cycle from CPU. Similarly an output word transfer from memory to the IOP is directed from the IOP to the output devices at device rate and capacity.

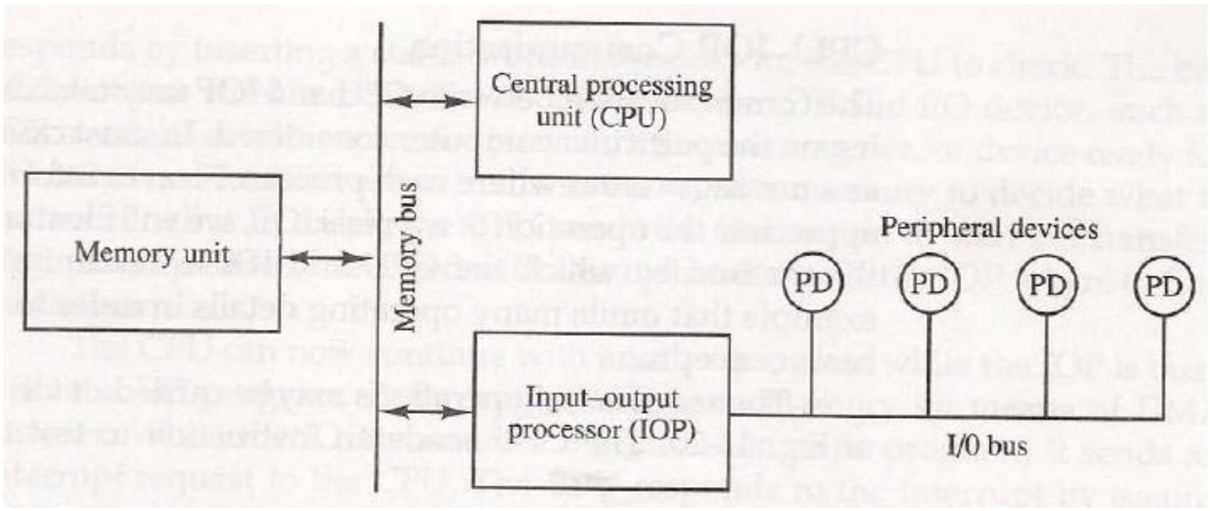


Figure: Block diagram of computer with I/O processor

CPU – IOP Communication

The communication between CPU and IOP may take different forms depending upon a particular computer considered. Here is the flow chart of CPU-IOP communication.

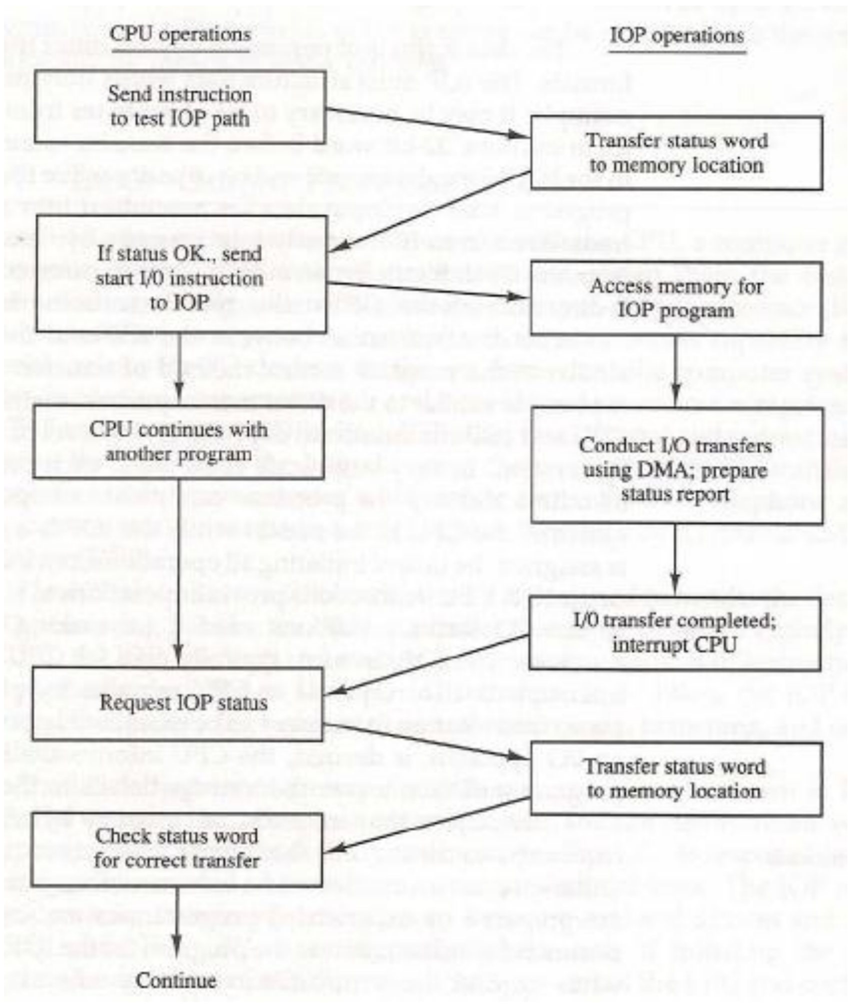

CPU operations
IOP operations
Send instruction to test IOP path
Transfer status word to memory location
If status OK., send start I/O instruction to IOP
Access memory for IOP program
CPU continues with another program
Conduct I/O transfers using DMA; prepare status report
I/0 transfer completed; interrupt CPU
Request IOP status
Transfer status word to memory location
Check status word for correct transfer
Continue

Figure: Flow chart of CPU-IOP Communication

4.5 External Interfaces

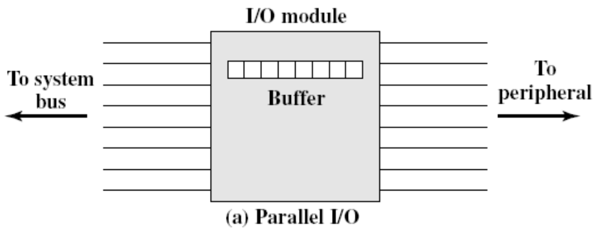
The interface to a peripheral from an I/O module must be tailored to the nature and operation of the peripheral. One major characteristic of the interface is whether it is serial or parallel. In a **parallel interface**, there are multiple lines connecting the I/O module and the peripheral, and multiple bits are transferred simultaneously. In a **serial interface**, there is only one line used to transmit data, and bits must be transmitted one at a time. A parallel interface has traditionally been used for higher-speed peripherals, such as tape and disk, while the serial interface has traditionally been used for printers and terminals. With a new generation of high-speed serial interfaces, parallel interfaces are becoming much less common. In either case, the I/O module must engage in a dialogue with the peripheral. In general terms, the dialogue for a write operation is as follows:

1. The I/O module sends a control signal requesting permission to send data.

2. The peripheral acknowledges the request.

3. The I/O module transfers data (one word or a block depending on the peripheral).

4. The peripheral acknowledges receipt of the data.



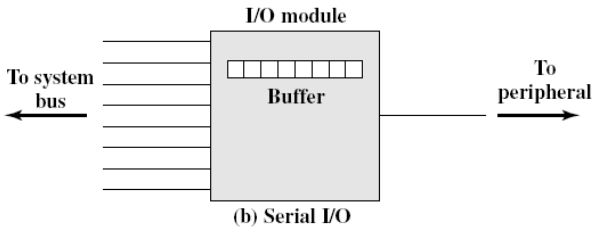


Figure: Parallel and Serial I/O

A read operation proceeds similarly. Key to the operation of an I/O module is an internal buffer that can store data being passed between the peripheral and the rest of the system. This buffer allows the I/O module to compensate for the differences in speed between the system bus and its external lines.

FireWire Serial Bus

With processor speeds reaching gigahertz range and storage devices holding multiple gigabits, the I/O demands for personal computers, workstations, and servers are formidable. Yet the high-speed I/O channel technologies that have been developed for mainframe and supercomputer systems are too expensive and bulky for use on these smaller systems. Accordingly, there has been great interest in developing a high-speed alternative to Small Computer System Interface (SCSI) and other small system I/O interfaces. The result is the IEEE standard 1394, for a High Performance Serial Bus, commonly known as FireWire.

InfiniBand

InfiniBand is a recent I/O specification aimed at the high-end server market. The first version of the specification was released in early 2001 and has attracted numerous vendors. The standard describes an architecture and specifications for data flow among processors and intelligent I/O devices. InfiniBand has become a popular interface for storage area networking and other large storage configurations. In essence, InfiniBand 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.

4.6 Data communication processor

A data communication processor is a processor in an I/O processor that distributes and collects data from many remote terminals connected through telephone or other communication lines. It is a specialized processor designed to communicate directly with data communication networks. With the use of data communication processor, the computer can service fragments of each networks demand in an interspersed manner and thus have apparent behavior of servicing many users at once. I/O processors communicate with the peripherals through a common I/O bus and use it to transfer the information to and from the I/O processor. A data communication processor communicates with each terminal through a single pair of wire. Both data and control information are transmitted in serial fashion that results in transfer rate much slower. The work of data communication processor is to transmit and collect digital information to and from each terminal, determine if information is data or control and respond to the request accordingly. The processor also must communicate with the CPU and memory in the same manner as any I/O process.

4.7 Buses

A bus is a communication pathway connecting two or more devices. Key characteristics are that it is a shared transmission medium.

A bus line may be classified as:

1. Address bus
2. Data bus
3. Control bus

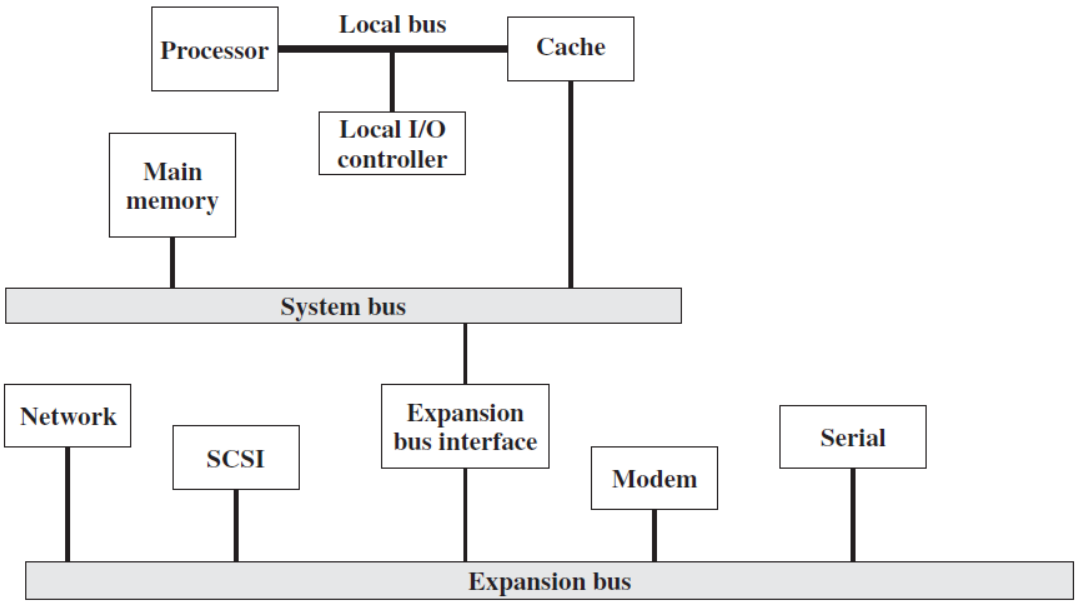
Buses may be serial or parallel, and the bus performance can be limited by:

1. Data propagation delay through longer buses.
2. Demand for access to the bus from all devices at the same time.

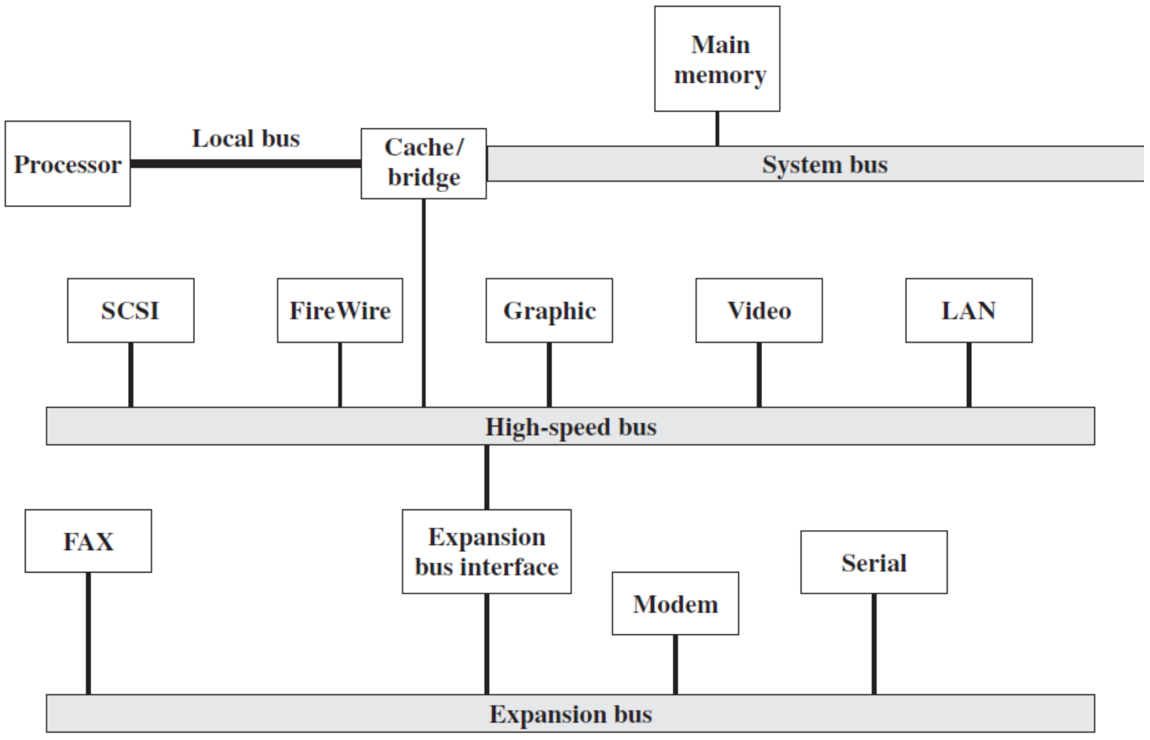
To avoid the bottle neck, multiple buses hierarchies are used. Their property include

- High speed limited access buses close to the processor.
- Slower speed general access buses further away.

Different bus hierarchy



(a) Traditional bus architecture



(b) High-performance architecture

a. Traditional Bus hierarchy:

- Local bus connects processor to cache, local I/O controller.
- May support one or more local devices.
- Processor connected to system via cache.
- An expansion bus connects other devices to system.

- An expansion bus interfaces buffers data transfer between the system buses and I/O controller on expansion bus.
- Network controller includes LANs, WANs, SCSI (Small Computer System Interface) that supports local disk drives and other peripherals.
- This traditional bus performance worsens when higher performance I/O devices are connected.

b. High performance architecture

- Local bus that connects the processor to a cache controller which in turn connected to a system bus that supports main memory.
- Cache controller is integrated into a bridge or buffering device that connects to high speed bus.
- High speed bus arrangements specifically designed to support high capacity I/O devices.
- Lower speed devices are still supported and connected to expansion buses connected to high speed bus through expansion bus interface.
- The advantage of this arrangement is that high speed bus brings high demand devices closer to the processor and at the same time is independent of the processor.

Elements of Bus Design

1. Types:
    a. Dedicated: Bus line that is permanently assigned to one function or to a physical subset of computer components.
    b. Multiplexed: Using same bus for multiple purposes. e.g. first sending address through the bus and making address valid line active, signal address is being transmitted. Then after using same bus to transmit data for read or write data transfer. The advantage will be, the use of fewer lines which saves space and usually cost. The disadvantage would be more complex circuitry needed and at certain events sharing or bus cannot be possible in parallel.

2. Method of arbitration
    - It is the process of insuring only one device places information onto the bus at a time.
    - In master slave mechanism
        - Purpose is to designate one device, either the processor or an I/O module as master.
        - Master: given control of the bus and can place information onto it.
        - Slave: receives information from master
    - Bus arbitration can be obtained in two methods:
    - Centralized: where central bus controller mediates all devices request for the bus.
    - De-centralized: where no centralized controller is used. All devices contain logic to control access to the bus instead.
3. Bus width
    The number of bits which are carried out simultaneously more, more width greater number of bits transfer.

4. Timing
    a. Synchronous timing: time controlled by a common clock signal. All events start at the beginning of a clock cycle, synchronous transfer are timed using a common clock signal
    b. Asynchronous timing: timing based on handshaking protocol. It is more flexible than synchronous bus but more complicated. It can accommodate wider range of devices speeds. Asynchronous transfers are timed with handshake signals.
5. Data transfer type
    a. Read: Reading data from memory, I/O.
    b. Write: Writing data to memory, I/O.
    c. Read-Modify-Write.
    d. Read-After-Write.
    e. Block: some buses support block data transfer.