UNIT-V

INPUT/OUTPUT ORGANIZATION

ACCESSING I/O-DEVICES

- A **single bus-structure** can be used for connecting I/O-devices to a computer (Figure 7.1).
- Each I/O device is assigned a unique set of address.
- Bus consists of 3 sets of lines to carry address, data & control signals.
- When processor places an address on address-lines, the intended-device responds to the command.
- The processor requests either a read or write-operation.
- The requested-data are transferred over the data-lines.

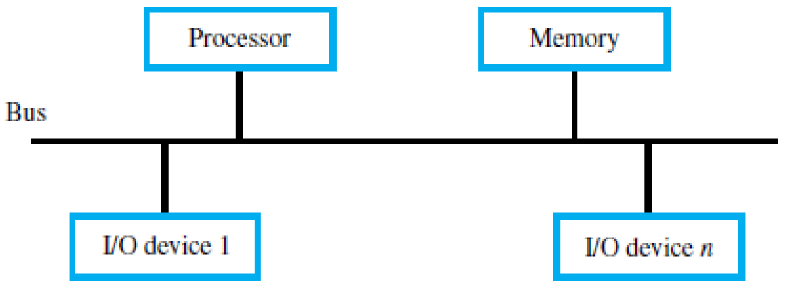


Figure 7.1 A single-bus structure.

- There are 2 ways to deal with I/O-devices: 1) Memory-mapped I/O & 2) I/O-mapped I/O.

1) Memory-Mapped I/O

- ➢ Memory and I/O-devices share a common address-space.
- ➢ Any data-transfer instruction (like Move, Load) can be used to exchange information.
- ➢ For example,

Move DATAIN, R0; This instruction sends the contents of location DATAIN to register R0.

Here, DATAIN address of the input-buffer of the keyboard.

2) I/O-Mapped I/O

- ➢ Memory and I/0 address-spaces are different.
- ➢ A special instructions named **IN** and **OUT** are used for data-transfer.
- ➢ Advantage of separate I/O space: I/O-devices deal with fewer address-lines.

I/O Interface for an Input Device

1) Address Decoder: enables the device to recognize its address when this address appears on the address-lines (Figure 7.2).

2) Status Register: contains information relevant to operation of I/O-device.

3) Data Register: holds data being transferred to or from processor. There are 2 types:

i) DATAIN Input-buffer associated with keyboard.

ii) DATAOUT Output data buffer of a display/printer.

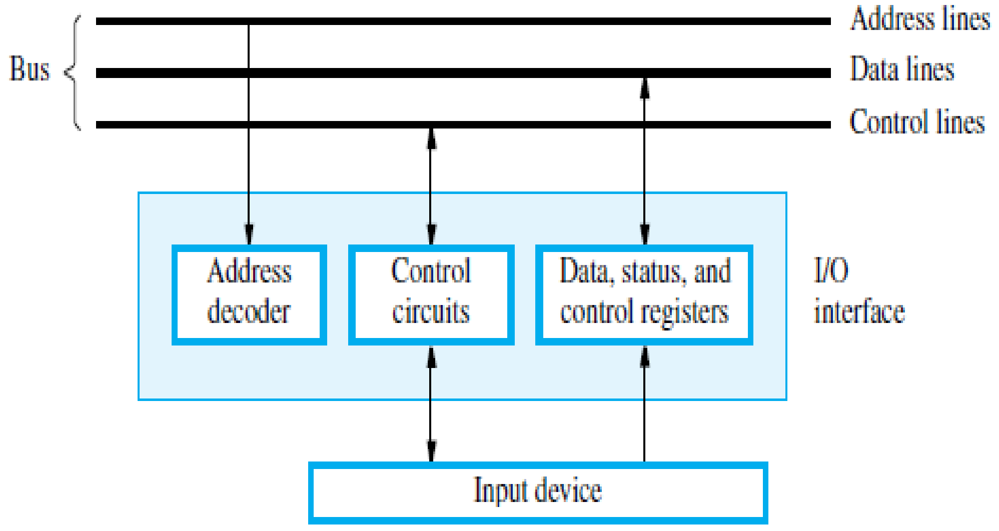


Figure 7.2 I/O interface for an input device.

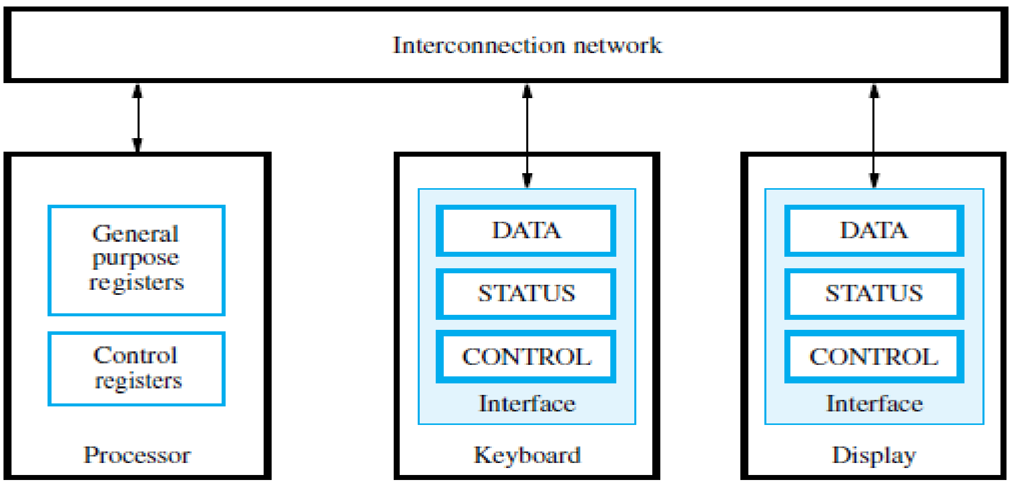


Figure 3.2 The connection for processor, keyboard, and display.

MECHANISMS USED FOR INTERFACING I/O-DEVICES

1) Program Controlled I/O

- Processor repeatedly checks status-flag to achieve required synchronization b/w processor & I/O device.

(We say that the processor polls the device).

- Main drawback:

  The processor wastes time in checking status of device before actual data-transfer takes place.

2) Interrupt I/O

- I/O-device initiates the action instead of the processor.
- I/O-device sends an INTR signal over bus whenever it is ready for a data-transfer operation.
- Like this, required synchronization is done between processor & I/O device.

3) Direct Memory Access (DMA)

- Device-interface transfer data directly to/from the memory w/o continuous involvement by the processor.
- DMA is a technique used for high speed I/O-device.

INTERRUPTS

- There are many situations where other tasks can be performed while waiting for an I/O device to become ready.
- A hardware signal called an Interrupt will alert the processor when an I/O device becomes ready.
- Interrupt-signal is sent on the interrupt-request line.
- The processor can be performing its own task without the need to continuously check the I/O-device.
- The routine executed in response to an interrupt-request is called ISR.
- The processor must inform the device that its request has been recognized by sending INTA signal.

  (INTR → Interrupt Request, INTA → Interrupt Acknowledge, ISR →Interrupt Service Routine)
- For example, consider COMPUTE and PRINT routines (Figure 3.6).

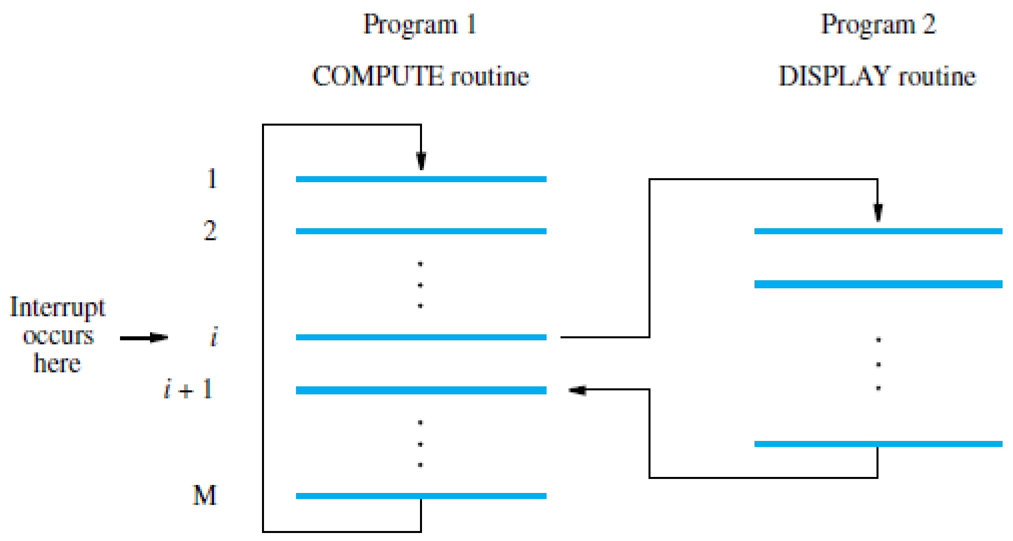


Figure 3.6 Transfer of control through the use of interrupts.

- The processor first completes the execution of instruction i.
- Then, processor loads the PC with the address of the first instruction of the ISR.
- After the execution of ISR, the processor has to come back to instruction i+1.
- Therefore, when an interrupt occurs, the current content of PC is put in temporary storage location.
- A return at the end of ISR reloads the PC from that temporary storage location.
- This causes the execution to resume at instruction i+1.
- When processor is handling interrupts, it must inform device that its request has been recognized.
- This may be accomplished by INTA signal.
- The task of saving and restoring the information can be done automatically by the processor.
- The processor saves only the contents of **PC & Status register.**
- Saving registers also increases the Interrupt Latency.
- **Interrupt Latency** is a delay between

→ time an interrupt-request is received and

→ start of the execution of the ISR.

- Generally, the long interrupt latency in unacceptable.

Difference between Subroutine & ISR MAIN

Subroutine	**ISR**
A subroutine performs a function required by the program from which it is called.	ISR may not have anything in common with program being executed at time INTR is received
Subroutine is just a linkage of 2 or more function related to each other.	Interrupt is A mechanism for coordinating I/O transfers.

INTERRUPT HARDWARE

- Most computers have several I/O devices that can request an interrupt.
- A single interrupt-request (IR) line may be used to serve n devices (Figure 4.6).
- All devices are connected to IR line via switches to ground.
- To request an interrupt, a device closes its associated switch.
- Thus, if all IR signals are inactive, the voltage on the IR line will be equal to V_{dd}.
- When a device requests an interrupt, the voltage on the line drops to 0.
- This causes the INTR received by the processor to go to 1.
- The value of INTR is the logical OR of the requests from individual devices.

$$\mathbf{INTR=INTR_1+ INTR_{2}+ .... +INTR_n}$$

- A special gates known as open-collector or open-drain are used to drive the INTR line.
- The Output of the open collector control is equal to a switch to the ground that is

  → open when gates input is in "0" state and

  → closed when the gates input is in "1" state.
- Resistor R is called a **Pull-up Resistor** because

  it pulls the line voltage up to the high-voltage state when the switches are open.

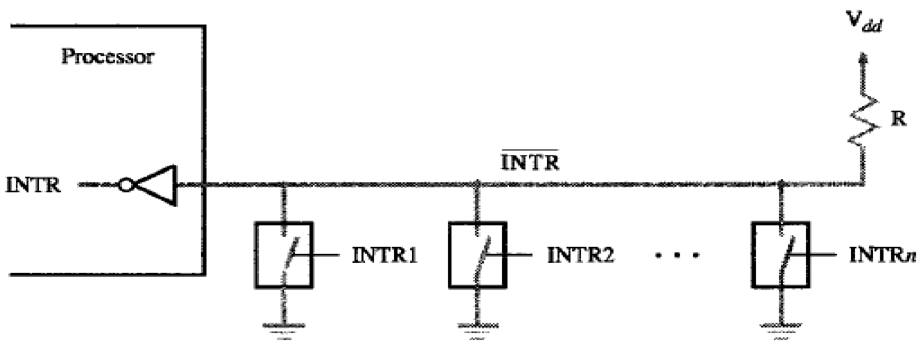


Figure 4.6 An equivalent circuit for an open-drain bus used to implement a common interrupt-request line.

ENABLING & DISABLING INTERRUPTS

- All computers fundamentally should be able to enable and disable interruptions as desired.
- The problem of infinite loop occurs due to successive interruptions of active INTR signals.
- There are 3 mechanisms to solve problem of infinite loop:
  1) Processor should ignore the interrupts until execution of first instruction of the ISR.
  2) Processor should automatically disable interrupts before starting the execution of the ISR.
  3) Processor has a special INTR line for which the interrupt-handling circuit.

     Interrupt-circuit responds only to leading edge of signal. Such line is called edge-triggered.
- Sequence of events involved in handling an interrupt-request:
  1) The device raises an interrupt-request.
  2) The processor interrupts the program currently being executed.
  3) Interrupts are disabled by changing the control bits in the processor status register (PS).
  4) The device is informed that its request has been recognized.

     In response, the device deactivates the interrupt-request signal.
  5) The action requested by the interrupt is performed by the interrupt-service routine.
  6) Interrupts are enabled and execution of the interrupted program is resumed.

HANDLING MULTIPLE DEVICES

- While handling multiple devices, the issues concerned are:
  1) How can the processor recognize the device requesting an interrupt?
  2) How can the processor obtain the starting address of the appropriate ISR?
  3) Should a device be allowed to interrupt the processor while another interrupt is being serviced?
  4) How should 2 or more simultaneous interrupt-requests be handled?

INTERRUPT NESTING

- A multiple-priority scheme is implemented by using separate INTR & INTA lines for each device
- Each INTR line is assigned a different priority-level (Figure 4.7).
- Priority-level of processor is the priority of program that is currently being executed.
- Processor accepts interrupts only from devices that have higher-priority than its own.
- At the time of execution of ISR for some device, priority of processor is raised to that of the device.
- Thus, interrupts from devices at the same level of priority or lower are disabled.

Privileged Instruction

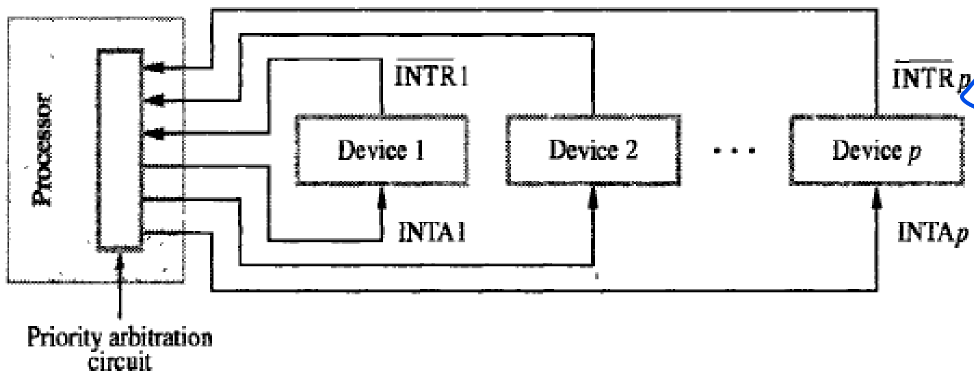
- Processor's priority is encoded in a few bits of PS word. (PS □ Processor-Status).
- Encoded-bits can be changed by **Privileged Instructions** that write into PS.
- Privileged-instructions can be executed only while processor is running in **Supervisor Mode**.
- Processor is in supervisor-mode only when executing operating-system routines.

Privileged Exception

- User program cannot

  → accidently or intentionally change the priority of the processor &

  → disrupt the system-operation.
- An attempt to execute a privileged-instruction while in user-mode leads to a **Privileged Exception**.


Processor
$\overline{\text{INTR}}1$
$\overline{\text{INTR}}p$
Device 1
Device 2
Device p
INTA1
INTAp
Priority arbitration circuit

Imp

Figure 4.7 Implementation of interrupt priority using individual interrupt-request and acknowledge lines.

SIMULTANEOUS REQUESTS

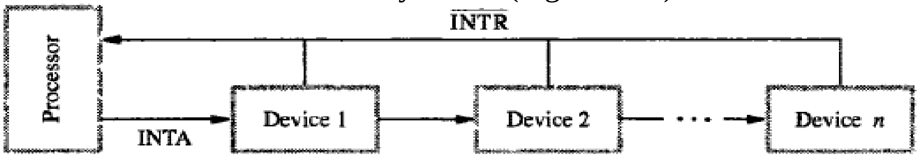
- The processor must have some mechanisms to decide which request to service when simultaneous requests arrive.
- INTR line is common to all devices (Figure 4.8a).
- INTA line is connected in a daisy-chain fashion.
- INTA signal propagates serially through devices.
- When several devices raise an interrupt-request, INTR line is activated.
- Processor responds by setting INTA line to 1. This signal is received by device 1.
- Device-1 passes signal on to device 2 only if it does not require any service.
- If device-1 has a pending-request for interrupt, the device-1

  → blocks INTA signal &

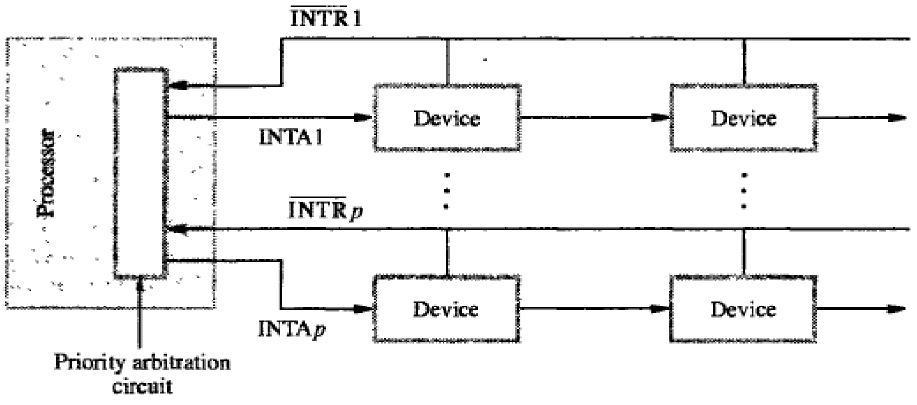
  → proceeds to put its identifying-code on data-lines.
- Device that is electrically closest to processor has highest priority.
- **Advantage:** It requires fewer wires than the individual connections.

Arrangement of Priority Groups

- Here, the devices are organized in groups & each group is connected at a different priority level.
- Within a group, devices are connected in a daisy chain. (Figure 4.8b).



(a) Daisy chain



(b) Arrangement of priority groups

Figure 4.8 Interrupt priority schemes.

DIRECT MEMORY ACCESS (DMA)

- The transfer of a block of data directly b/w an external device & main-memory w/o continuous involvement by processor is called DMA.
- DMA controller

  → is a control circuit that performs DMA transfers (Figure 8.13).

  → is a part of the I/O device interface.

  → performs the functions that would normally be carried out by processor.
- While a DMA transfer is taking place, the processor can be used to execute another program.

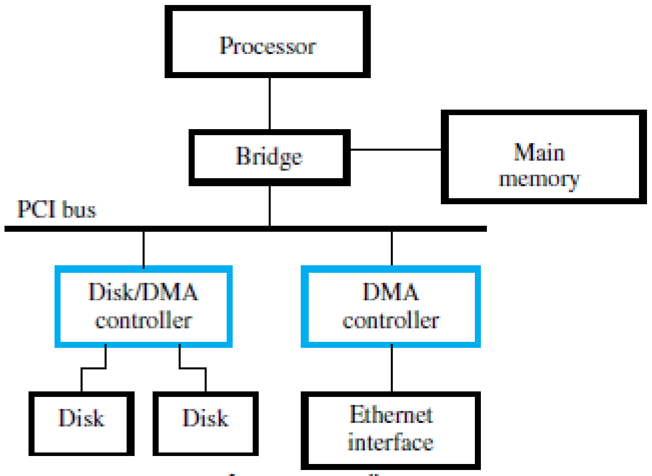


Figure 8.13 Use of DMA controllers in a computer system.

- DMA interface has three registers (Figure 8.12):

  1) First register is used for storing starting-address.

  2) Second register is used for storing word-count.

  3) Third register contains status- & control-flags.

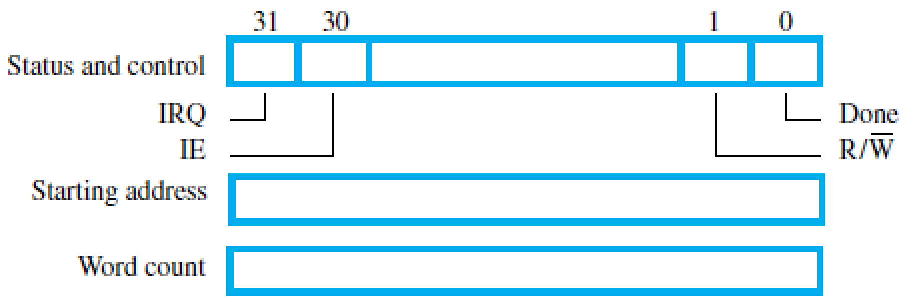


Figure 8.12 Typical registers in a DMA controller.

- The R/W bit determines direction of transfer.

  If R/W=1, controller performs a read-operation (i.e. it transfers data from memory to I/O),

  Otherwise, controller performs a write-operation (i.e. it transfers data from I/O to memory).

- If Done=1, the controller

  → has completed transferring a block of data and

  → is ready to receive another command. (IE→ Interrupt Enable).

- If IE=1, controller raises an interrupt after it has completed transferring a block of data.
- If IRQ=1, controller requests an interrupt.
- Requests by DMA devices for using the bus are always given higher priority than processor requests.
- There are 2 ways in which the DMA operation can be carried out:

  1) Processor originates most memory-access cycles.

  ➢ DMA controller is said to "steal" memory cycles from processor.

  ➢ Hence, this technique is usually called **Cycle Stealing**.

  2) DMA controller is given exclusive access to main-memory to transfer a block of data without any interruption. This is known as **Block Mode** (or burst mode).

BUS ARBITRATION = Process of selecting bus master

- The device that is allowed to initiate data-transfers on bus at any given time is called **bus-master**.
- There can be only one bus-master at any given time.
- **Bus Arbitration** is the process by which

  → next device to become the bus-master is selected &

  → bus-mastership is transferred to that device.

- The two approaches are:

  1) Centralized Arbitration: A single bus-arbiter performs the required arbitration.

  2) Distributed Arbitration: All devices participate in selection of next bus-master.

- A conflict may arise if both the processor and a DMA controller or two DMA controllers try to use the bus at the same time to access the main-memory.
- To resolve this, an arbitration procedure is implemented on the bus to coordinate the activities of all devices requesting memory transfers.
- The bus arbiter may be the processor or a separate unit connected to the bus.

CENTRALIZED ARBITRATION

- A single bus-arbiter performs the required arbitration (Figure: 4.20).
- Normally, processor is the bus-master.
- Processor may grant bus-mastership to one of the DMA controllers.
- A DMA controller indicates that it needs to become bus-master by activating BR line.
- The signal on the BR line is the logical OR of bus-requests from all devices connected to it.
- Then, processor activates BG1 signal indicating to DMA controllers to use bus when it becomes free.
- BG1 signal is connected to all DMA controllers using a daisy-chain arrangement.
- If DMA controller-1 is requesting the bus,

  Then, DMA controller-1 blocks propagation of grant-signal to other devices. Otherwise,

  DMA controller-1 passes the grant downstream by asserting BG2.
- Current bus-master indicates to all devices that it is using bus by activating BBSY line.
- The bus-arbiter is used to coordinate the activities of all devices requesting memory transfers.
- Arbiter ensures that only 1 request is granted at any given time according to a priority scheme. (BR

   Bus-Request, BG Bus-Grant, BBSY Bus Busy).

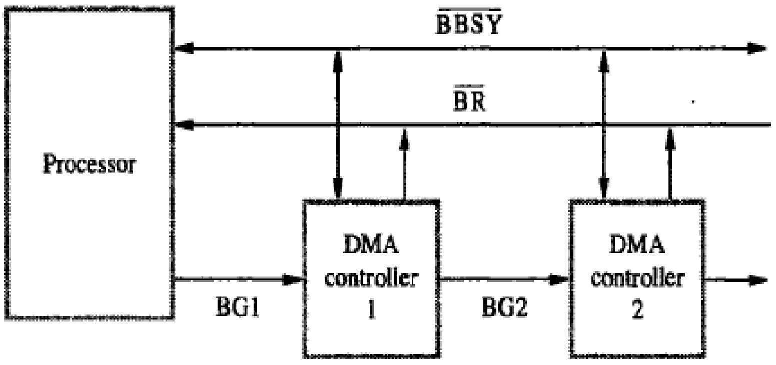


Figure 4.20 A simple arrangement for bus arbitration using a daisy chain.

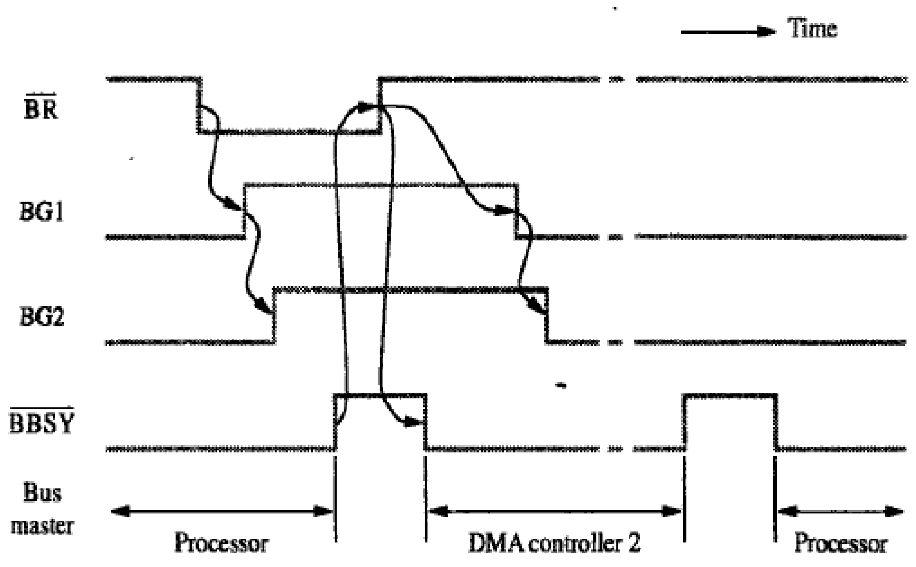


Figure 4.21 Sequence of signals during transfer of bus mastership for the devices in Figure 4.20.

- The timing diagram shows the sequence of events for the devices connected to the processor.
- DMA controller-2

  → requests and acquires bus-mastership and

  → later releases the bus. (Figure: 4.21).
- After DMA controller-2 releases the bus, the processor resources bus-mastership.

DISTRIBUTED ARBITRATION

- All device participate in the selection of next bus-master (Figure 4.22).
- Each device on bus is assigned a 4-bit identification number (ID).

Advantage:

This approach offers higher reliability since operation of bus is not dependent on any single device.

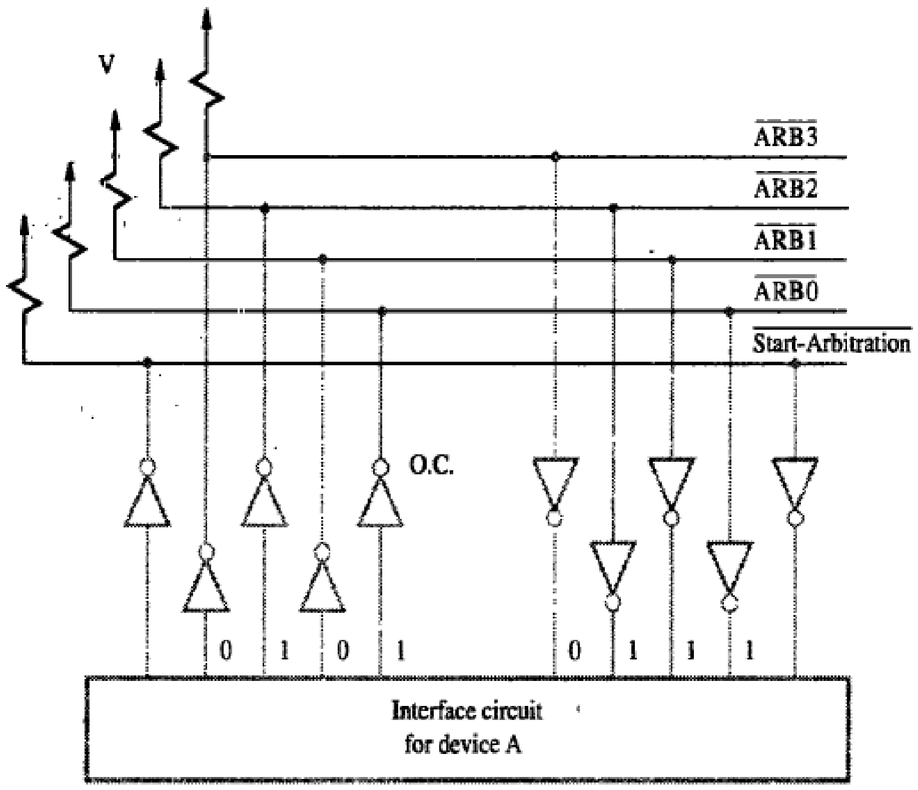


Figure 4.22 A distributed arbitration scheme.

For E.g.

- Assume 2 devices A & B have their ID 5 (0101), 6 (0110) and their code is 0111.
- Each device compares the pattern on the arbitration line to its own ID starting from MSB.
- If the device detects a difference at any bit position, it disables the drivers at that bit position.
- Driver is disabled by placing ”0” at the input of the driver.
- In e.g. “A” detects a difference in line ARB1, hence it disables the drivers on lines ARB1 & ARB0.
- This causes pattern on arbitration-line to change to 0110. This means that “B” has won contention.

BUS

- Bus

→ is used to inter-connect main-memory, processor & I/O-devices

→ includes lines needed to support interrupts & arbitration.

- Primary function: To provide a communication-path for transfer of data.
- **Bus protocol** is set of rules that govern the behavior of various devices connected to the buses.
- Bus-protocol specifies parameters such as:

  → asserting control-signals

  → timing of placing information on bus

  → rate of data-transfer.

- A typical bus consists of 3 sets of lines:
  1) Address,
  2) Data &
  3) Control lines.
- Control-signals

  → specify whether a read or a write-operation is to be performed.

  → carry timing information i.e. they specify time at which I/O-devices place data on the bus.

- R/W line specifies

  → read-operation when R/W=1.

  → write-operation when R/W=0.

- During data-transfer operation,
  - ➢ One device plays the role of a bus-master.
  - ➢ Master-device initiates the data-transfer by issuing read/write command on the bus.
  - ➢ The device addressed by the master is called as Slave.
- Two types of Buses: 1) Synchronous and 2) Asynchronous.

INTERFACE-CIRCUITS

- An **I/O Interface** consists of the circuitry required to connect an I/O device to a computer-bus.
- On one side of the interface, we have bus signals.

  On the other side, we have a data path with its associated controls to transferdata between the interface and the I/O device known as **port**.
- Two types are:

  1. Parallel Port transfers data in the form of a number of bits (8 or 16) simultaneously to or from the device.

  2. Serial Port transmits and receives data one bit at a time.
- Communication with the bus is the same for both formats.
- The conversion from the parallel to the serial format, and vice versa, takes place inside the interface-circuit.
- In parallel-port, the connection between the device and the computer uses

  → a multiple-pin connector and

  → a cable with as many wires.
- This arrangement is suitable for devices that are physically close to the computer.
- In serial port, it is much more convenient and cost-effective where longer cables are needed.

Functions of I/O Interface

1) Provides a storage buffer for at least one word of data.
2) Contains status-flags that can be accessed by the processor to determine whether the buffer is full or empty.
3) Contains address-decoding circuitry to determine when it is being addressed by the processor.
4) Generates the appropriate timing signals required by the bus control scheme.
5) Performs any format conversion that may be necessary to transfer data between the bus and the I/O device (such as parallel-serial conversion in the case of a serial port).

PARALLEL-PORT

KEYBOARD INTERFACED TO PROCESSOR

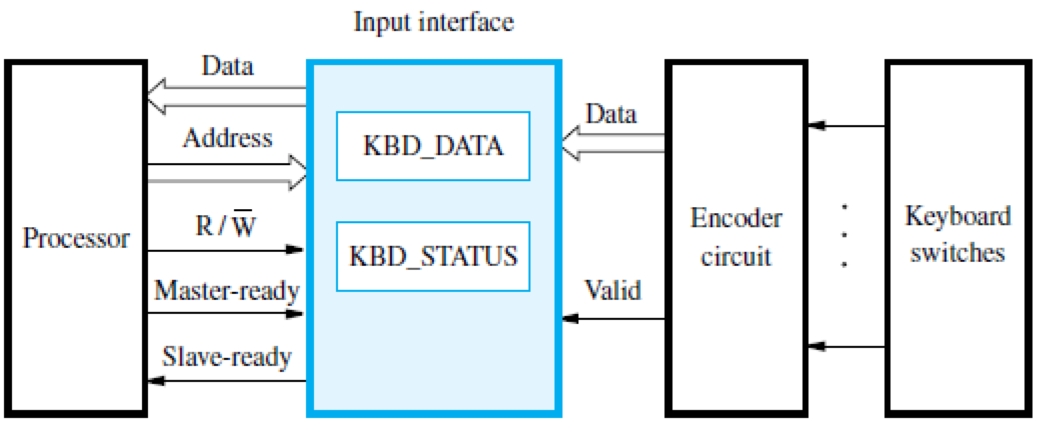


Figure 7.10 Keyboard to processor connection.

- The output of the encoder consists of
  - → bits representing the encoded character and
  - → one signal called **valid**, which indicates the key is pressed.
- The information is sent to the interface-circuits (Figure 7.10).
- Interface-circuits contain
  1) Data register DATAIN &
  2) Status-flag SIN.
- When a key is pressed, the Valid signal changes from 0 to1.

  Then, SIN=1 when ASCII code is loaded into DATAIN.

  SIN = 0 when processor reads the contents of the DATAIN.
- The interface-circuit is connected to the asynchronous bus.
- Data transfers on the bus are controlled using the handshake signals:
  1) Master ready &
  2) Slave ready.

INPUT-INTERFACE-CIRCUIT

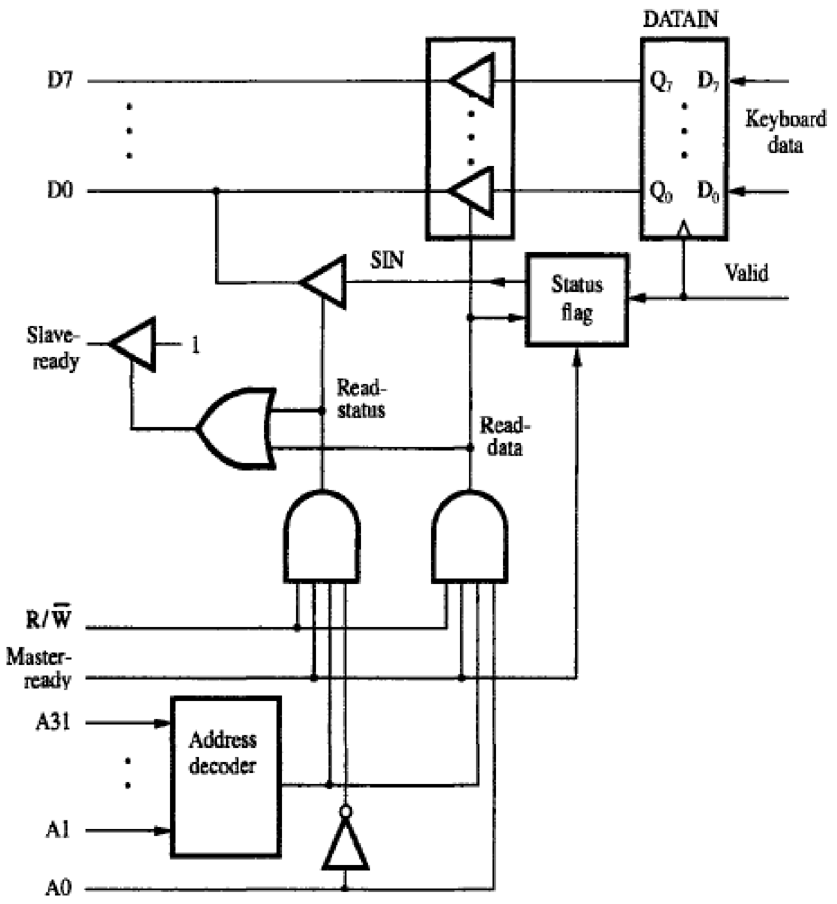


Figure 4.29: Input-interface-circuit

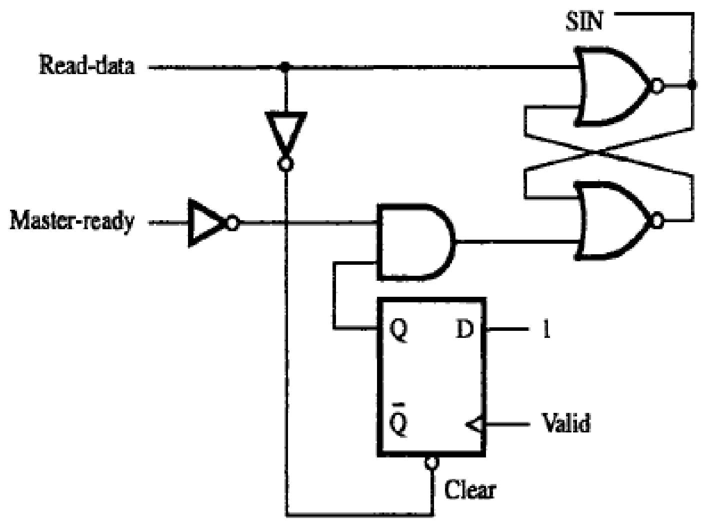


Figure 4.30 Circuit for the status flag block in Figure 4.29.

- Output-lines of DATAIN are connected to the data-lines of bus by means of 3-state drivers (Fig 4.29).
- Drivers are turned on when

  → processor issues a read signal and

  → address selects DATAIN.

• SIN signal is generated using a status-flag circuit (Figure 4.30).

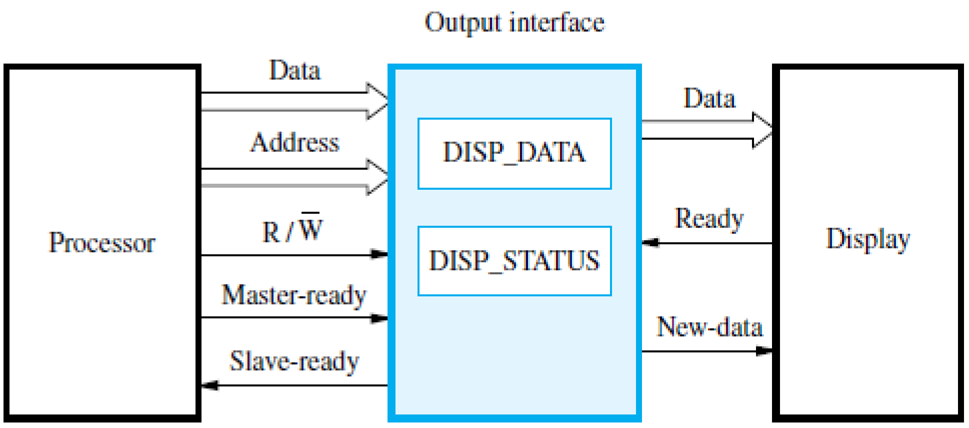


Figure 7.13 Display to processor connection.

SIN signal is connected to line D_0 of the processor-bus using a 3-state driver.

• Address-decoder selects the input-interface based on bits A_1 through A_{31}.

• Bit A_0 determines whether the status or data register is to be read, when Master-ready is active.

• Processor activates the Slave-ready signal, when either the Read-status or Read-data is equal to 1.

PRINTER INTERFACED TO PROCESSOR

• Keyboard is connected to a processor using a parallel-port.

• Processor uses

→ memory-mapped I/O and

→ asynchronous bus protocol.

• On the processor-side of the interface, we have:

→ Data-lines

→ Address-lines

→ Control or R/W line

→ Master-Ready signal and

→ Slave-Ready signal.

• On the keyboard-side of the interface, we have:

→ Encoder-circuit which generates a code for the key pressed.

→ Debouncing-circuit which eliminates the effect of a key.

→ Data-lines which contain the code for the key.

→ Valid line changes from 0 to 1 when the key is pressed. This causes the code to be loaded into DATAIN and SIN to be set to 1.

STANDARD I/O INTERFACE

- Consider a computer system using different interface standards.
- Let us look in to Processor bus and Peripheral Component Interconnect (PCI) bus (Figure 4.38).
- These two buses are interconnected by a circuit called **Bridge**.
- The bridge translates the signals and protocols of one bus into another.
- The bridge-circuit introduces a small delay in data transfer between processor and the devices.

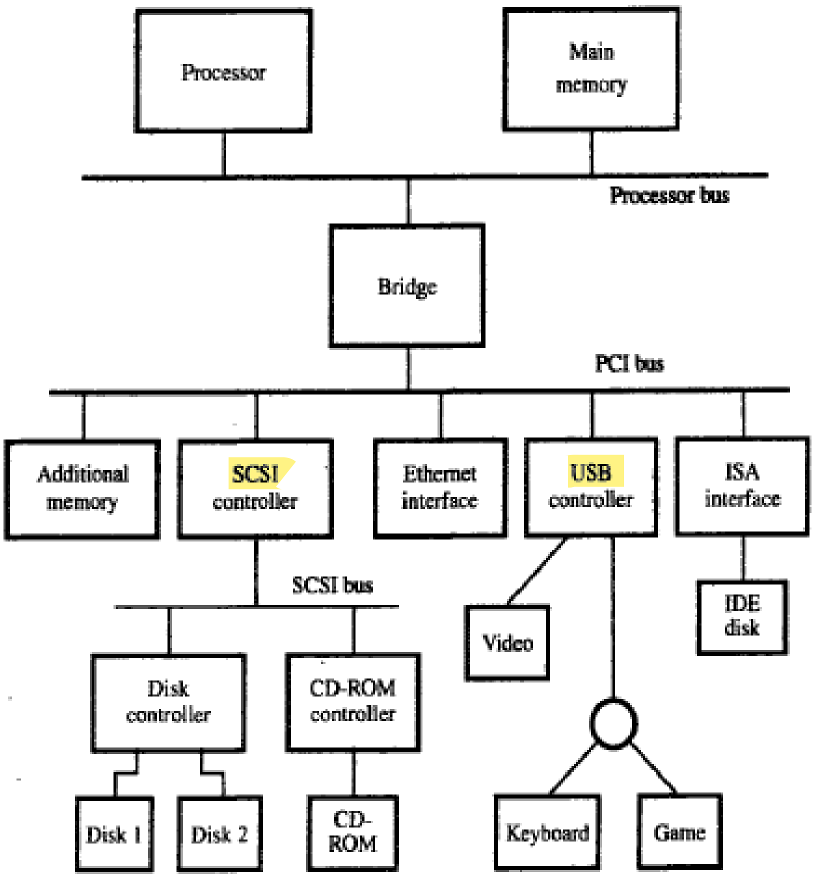


Figure 4.38 An example of a computer system using different interface standards.

The 3 major standard I/O interfaces are:

1) PCI (Peripheral Component Interconnect)
2) SCSI (Small Computer System Interface)
3) USB (Universal Serial Bus)

- PCI defines an expansion bus on the motherboard.
- SCSI and USB are used for connecting additional devices both inside and outside the computer-box.
- SCSI bus is a high speed parallel bus intended for devices such as disk and video display.
- USB uses a serial transmission to suit the needs of equipment ranging from keyboard to game control to internal connection.
- IDE (Integrated Device Electronics) disk is compatible with ISA which shows the connection to an Ethernet.

PCI

- PCI is developed as a low cost bus that is truly processor independent.
- PCI supports high speed disk, graphics and video devices.
- PCI has plug and play capability for connecting I/O devices.
- To connect new devices, the user simply connects the device interface board to the bus.

DATA TRANSFER IN PCI

- The data are transferred between cache and main-memory.
- The data is a sequence of words which are stored in successive memory-locations.
- During **read-operation**,
  - ➢ When the processor specifies an address, the memory responds by sending a sequence of data-words from successive memory-locations.
- During **write-operation**,
  - ➢ When the processor sends an address, a sequence of data-words is written into successive memory-locations.
- PCI supports read and write-operation.
- A read/write-operation involving a single word is treated as a burst of length one.
- PCI has 3 address-spaces. They are
  1) Memory address-space   M I/O C
  2) I/O address-space &
  3) Configuration address-space.
- I/O Address-space Intended for use with processor.

  Configuration space Intended to give PCI, its plug and play capability.
- **PCI Bridge** provides a separate physical connection to main-memory.
- The master maintains the address information on the bus until data-transfer is completed.
- At any time, only one device acts as **Bus-Master**.
- A master is called "initiator" which is either processor or DMA.
- The addressed-device that responds to read and write commands is called a **Target.**
- A complete transfer operation on the bus, involving an address and burst of data is called a **transaction.**

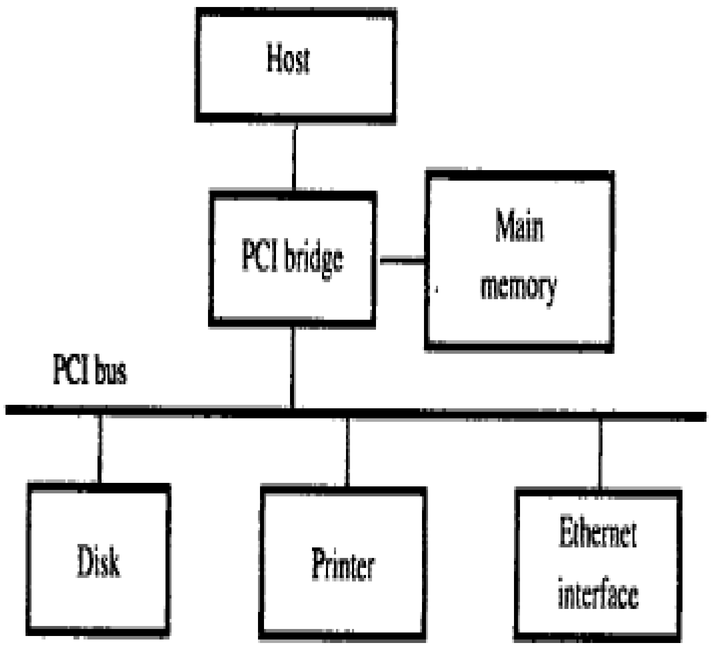


Figure 4.39 Use of a PCI bus in a computer system.

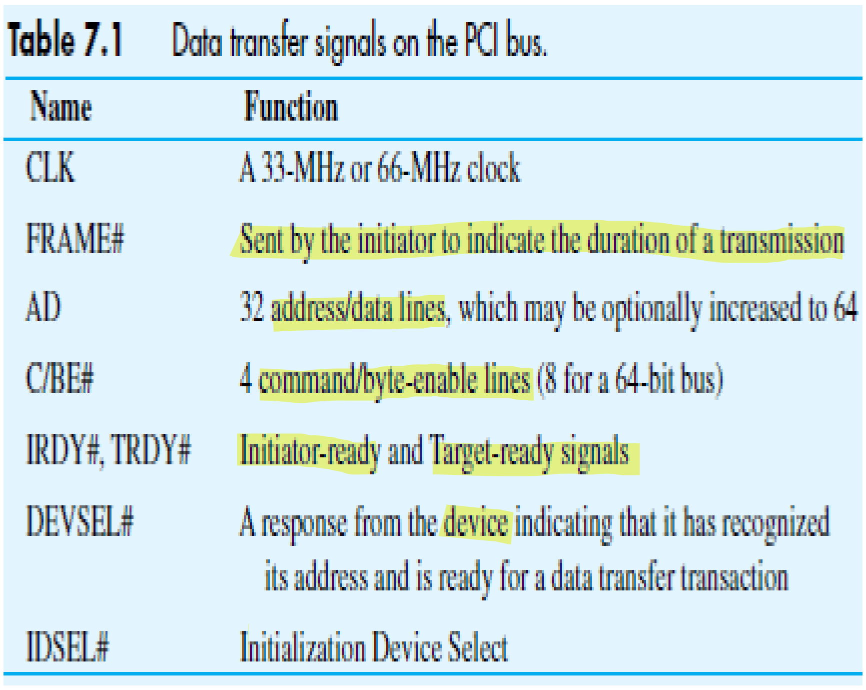


Table 7.1 Data transfer signals on the PCI bus.

Name	Function
CLK	A 33-MHz or 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transmission
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 command/byte-enable lines (8 for a 64-bit bus)
IRDY#, TRDY#	Initiator-ready and Target-ready signals
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction
IDSEL#	Initialization Device Select

SCSI Bus

- SCSI stands for Small Computer System Interface.
- SCSI refers to the standard bus which is defined by ANSI (American National Standard Institute).
- SCSI buses the several options. It may be,

Narrow bus	It has 8 data-lines & transfers 1 byte at a time.
Wide bus	It has 16 data-lines & transfer 2 byte at a time.
Single-Ended Transmission	Each signal uses separate wire.
HVD (High Voltage Differential)	It was 5v (TTL cells)
LVD (Low Voltage Differential)	It uses 3.3v

- Because of these various options, SCSI connector may have 50, 68 or 80 pins. The data transfer rate ranges from 5MB/s to 160MB/s 320Mb/s, 640MB/s. The transfer rate depends on,

    1) Length of the cable

    2) Number of devices connected.

- To achieve high transfer rate, the bus length should be 1.6m for SE signaling and 12m for LVD signaling.
- The SCSI bus us connected to the processor-bus through the SCSI controller. The data are stored on a disk in blocks called sectors.

Each sector contains several hundreds of bytes. These data will not be stored in contiguous memory-location.

- SCSI protocol is designed to retrieve the data in the first sector or any other selected sectors.
- Using SCSI protocol, the burst of data are transferred at high speed.
- The controller connected to SCSI bus is of 2 types. They are1) Initiator * 2) Target

1) Initiator

➢ It has the ability to select a particular target & to send commands specifying the operation to be performed.

➢ They are the controllers on the processor side.

2) Target

➢ The disk controller operates as a target.

➢ It carries out the commands it receive from the initiator.

➢ The initiator establishes a logical connection with the intended target

Steps for Read-operation

1) The SCSI controller contends for control of the bus (initiator).

2) When the initiator wins the arbitration-process, the initiator

→ selects the target controller and

→ hands over control of the bus to it.

3) The target starts an output operation. The initiator sends a command specifying the required read-operation.

4) The target

→ sends a message to initiator indicating that it will temporarily suspend connection b/w them.

→ then releases the bus.

5) The target controller sends a command to the disk drive to move the read head to the first sector involved in the requested read-operation.

6. The target

→ transfers the contents of the data buffer to the initiator and

→ then suspends the connection again.

7) The target controller sends a command to the disk drive to perform another seek operation.

8) As the initiator controller receives the data, it stores them into the main-memory using the DMA approach.

9) The SCSI controller sends an interrupt to the processor indicating that the data are now available.

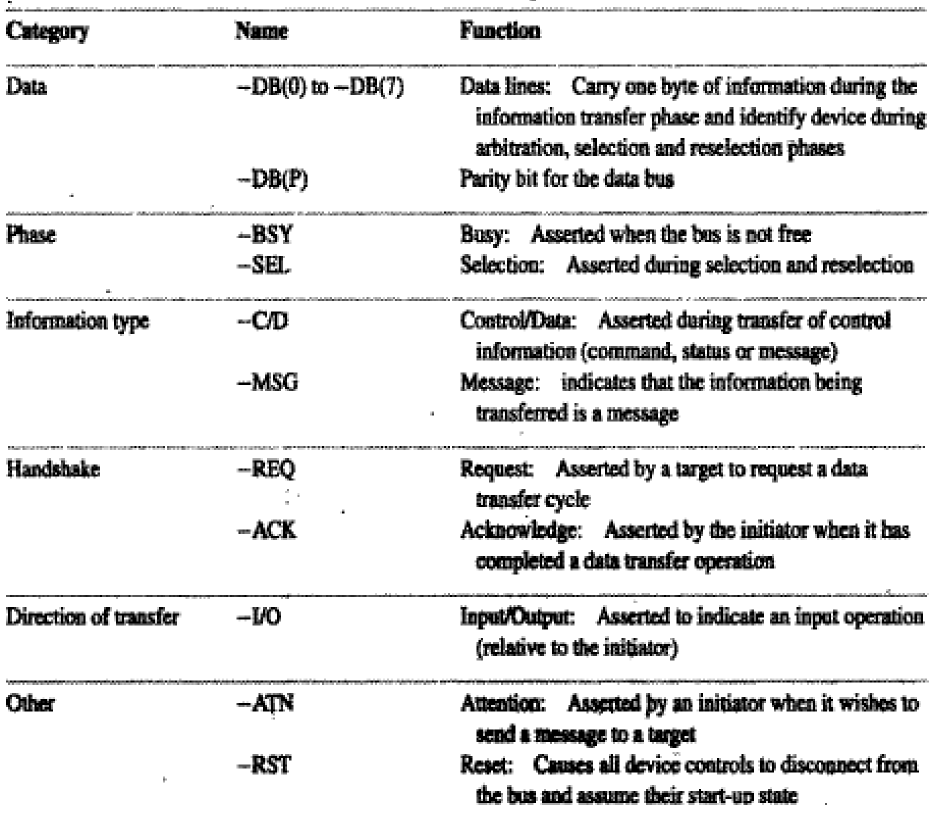
BUS SIGNALS OF SCSI

• The bus has no address-lines. Instead, it has data-lines to identify the bus-controllers involved in the selection/reselection/arbitration-process.

• For narrow bus, there are 8 possible controllers numbered from 0 to 7. For a wide bus, there are 16 controllers.

• Once a connection is established b/w two controllers, there is no further need for addressing & the data-lines are used to carry the data.

Table 4.4 The SCSI bus signals



Category	Name	Function
Data	–DB(0) to –DB(7)	Data lines: Carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	–DB(P)	Parity bit for the data bus
Phase	–BSY	Busy: Asserted when the bus is not free
	–SEL	Selection: Asserted during selection and reselection
Information type	–C/D	Control/Data: Asserted during transfer of control information (command, status or message)
	–MSG	Message: indicates that the information being transferred is a message
Handshake	–REQ	Request: Asserted by a target to request a data transfer cycle
	–ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction of transfer	–I/O	Input/Output: Asserted to indicate an input operation (relative to the initiator)
Other	–ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	–RST	Reset: Causes all device controls to disconnect from the bus and assume their start-up state

- All signal names are proceeded by minus sign.
- This indicates that the signals are active or that the data-line is equal to 1, when they are in the low voltage state.

PHASES IN SCSI BUS

- The phases in SCSI bus operation are:
  1) Arbitration
  2) Selection
  3) Information transfer
  4) Reselection

1) Arbitration

- When the –BSY signal is in inactive state,

  → the bus will be free & any controller can request the use of bus.
- SCSI uses distributed arbitration scheme because each controller may generate requests at the same time.
- Each controller on the bus is assigned a fixed priority.
- When –BSY becomes active, all controllers that are requesting the bus

  → examines the data-lines &

  → determine whether highest priority device is requesting bus at the same time.

- The controller using the highest numbered line realizes that it has won the arbitration-process.
- At that time, all other controllers disconnect from the bus & wait for –BSY to become inactive again.

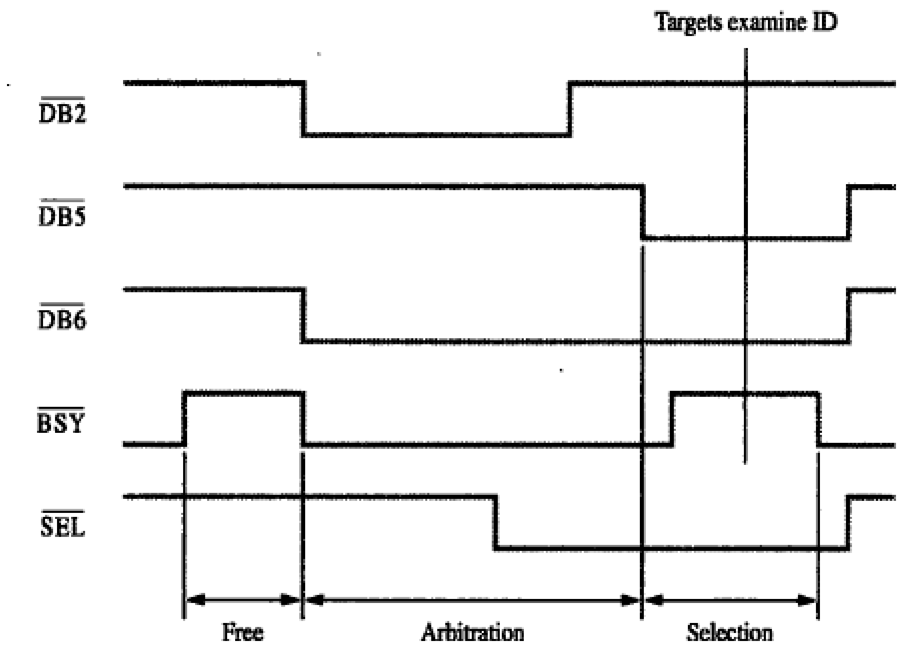


Figure 4.42 Arbitration and selection on the SCSI bus. Device 6 wins arbitration and selects device 2.

2) Information Transfer

- The information transferred between two controllers may consist of

  → commands from the initiator to the target

  → status responses from the target to the initiator or

  → data-transferred to/from the I/0 device.

- Handshake signaling is used to control information transfers, with the target controller taking the role of the bus-master.

3) Selection

- Here, Device
  → wins arbitration and

  → asserts –BSY and –DB6 signals.

- The Select Target Controller responds by asserting –BSY.
- This informs that the connection that it requested is established.

4) Reselection

- The connection between the two controllers has been reestablished, with the target in control of the bus as required for data transfer to proceed.

USB

- USB stands for Universal Serial Bus.
- USB supports 3 speed of operation. They are,
  1) Low speed (1.5 Mbps)
  2) Full speed (12 mbps) &
  3) High speed (480 mbps).

- The USB has been designed to meet the key objectives. They are,
  1) Provide a simple, low-cost and easy to use interconnection system.

     This overcomes difficulties due to the limited number of I/O ports available on a computer.
  2) Accommodate a wide range of data transfer characteristics for I/O devices.

     For e.g. telephone and Internet connections
  3) Enhance user convenience through a “plug-and-play” mode of operation.
- **Advantage:** USB helps to add many devices to a computer system at any time without opening the computer-box.

Port Limitation

- ➢ Normally, the system has a few limited ports.
- ➢ To add new ports, the user must open the computer-box to gain access to the internal expansion bus & install a new interface card.
- ➢ The user may also need to know to configure the device & the s/w.

Plug & Play

- ➢ The main objective: USB provides a plug & play capability.
- ➢ The plug & play feature enhances the connection of new device at any time, while the system is operation.
- ➢ The system should

  → Detect the existence of the new device automatically.

  → Identify the appropriate device driver s/w.

  → Establish the appropriate addresses.

  → Establish the logical connection for communication.

DEVICE CHARACTERISTICS OF USB

- The kinds of devices that may be connected to a computer cover a wide range of functionality.
- The speed, volume & timing constrains associated with data transfer to & from devices varies significantly.

Eg: 1 Keyboard

- Since the event of pressing a key is not synchronized to any other event in a computer system, the data generated by keyboard are called asynchronous.

  The data generated from keyboard depends upon the speed of the human operator which is about 100 bytes/sec.

Eg: 2 Microphone attached in a computer system internally/externally

- The sound picked up by the microphone produces an analog electric signal, which must be converted into digital form before it can be handled by the computer.
- This is accomplished by sampling the analog signal periodically.
- The sampling process yields a continuous stream of digitized samples that arrive at regular intervals, synchronized with the sampling clock. Such a stream is called isochronous (i.e.) successive events are separated by equal period of time.
- If the sampling rate in 'S' samples/sec then the maximum frequency captured by sampling process is s/2.
- A standard rate for digital sound is 44.1 KHz.

USB ARCHITECTURE

- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure as shown in the figure 7.17.
- Each node of the tree has a device called a **Hub**.
- A hub acts as an intermediate control point between the host and the I/O devices.
- At the root of the tree, a **Root Hub** connects the entire tree to the host computer.
- The leaves of the tree are the I/O devices being served (for example, keyboard or speaker).
- A hub copies a message that it receives from its upstream connection to all its downstream ports.
- As a result, a message sent by the host computer is broadcast to all I/O devices, but only the addressed-device will respond to that message.

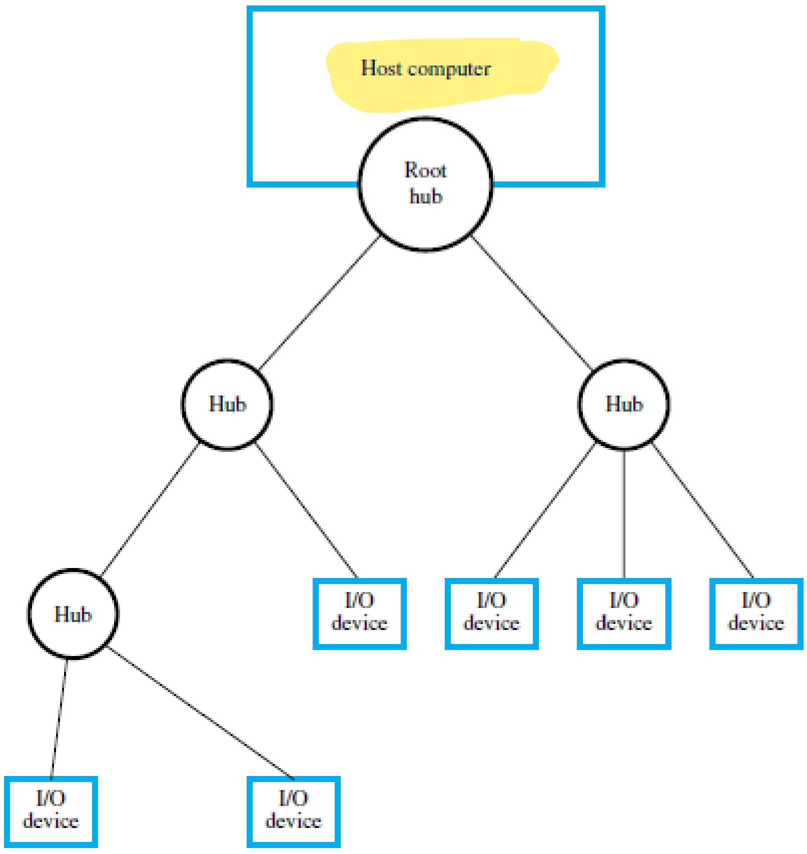


Figure 7.17 Universal Serial Bus tree structure.

USB ADDRESSING

- Each device may be a hub or an I/O device.
- Each device on the USB is assigned a 7-bit address.
- This address

  → is local to the USB tree and

  → is not related in any way to the addresses used on the processor-bus.
- A hub may have any number of devices or other hubs connected to it, and addresses are assigned arbitrarily.
- When a device is first connected to a hub, or when it is powered-on, it has the address 0.
- The hardware of the hub detects the device that has been connected, and it records this fact as part of its own status information.
- Periodically, the host polls each hub to

  → collect status information and

  → learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it uses sequence of commands to

  → send a reset signal on the corresponding hub port.

  - → read information from the device about its capabilities.
  - → send configuration information to the device, and
  - → assign the device a unique USB address.

- Once this sequence is completed, the device
  - → begins normal operation and
  - → responds only to the new address.