UNIT V

I/O Organization

Accessing I/O devices, Interrupts, Direct Memory Access , Buses, Interface circuits, Standard I/O Interfaces (PCI, SCSI, USB).

5.1 I/O ORGANIZATION

5.1.1 Accessing I/O Devices

Q1. Explain about the I/O devices and how to access them.

Ans : **(Imp.)**

- Input-output (I/O) devices vary substantially in their characteristics. One distinguishing factor among input devices (and also among output devices) is their data processing rate, defined as the average number of characters that can be processed by a device per second.
- Striking a character on the keyboard of a computer will cause a character (in the form of an ASCII code) to be sent to the computer. The amount of time passed before the next character is sent to the computer will depend on the skill of the user and even sometimes on his/her speed of thinking. It is often the case that the user knows what he/she wants to input, but sometimes they need to think before touching the next button on the keyboard. Therefore, input from a keyboard is slow and burst in nature and it will be a waste of time for the computer to spend its valuable time waiting for input from slow input devices. A mechanism is therefore needed whereby a device will have to interrupt the processor asking for attention whenever it is ready. This is called interrupt-driven communication between the computer and I/O devices .
- Consider the case of a disk. A typical disk should be capable of transferring data at rates exceeding several million bytes/second. It would be a waste of time to transfer data byte by byte or even word by word. Therefore, it is always the case that data is transferred in the form of blocks, that is, entire programs. It is also necessary to provide a mechanism that allows a disk to transfer this huge volume of data without the intervention of the CPU. This will allow the CPU to perform other useful operation(s) while a huge amount of data is being transferred between the disk and the memory.
- Figure (a) shows a simple arrangement for connecting the processor and the memory in a given computer system to an input device, for example, a keyboard and an output device such as a graphic display. A single bus consisting of the required address, data, control lines is used to connect the system's components in Figure (a).
- We are here concerned with the way the processor and the I/O devices exchange data. It has been indicated in the introduction part that there exists a big difference in the rate at which a processor can process information and those of input and output devices. One simple way to accommodate this speed difference is to have the input device, for example, a keyboard, deposit the character struck by the user in a register (input register), which indicates the availability of that character to the processor. When the input character has been taken by the processor, this will be indicated to the input device in order to proceed and input the next character, and so on. Similarly, when the

processor has a character to output (display), it deposits it in a specific register dedicated for communication with the graphic display (output register). When the character has been taken by the graphic display, this will be indicated to the processor such that it can proceed and output the next character, and so on. This simple way of communication between the processor and I/O devices, called I/O protocol, requires the availability of the input and output registers. In a typical computer system, there is a number of input registers, each belonging to a specific input device. There is also a number of output registers,

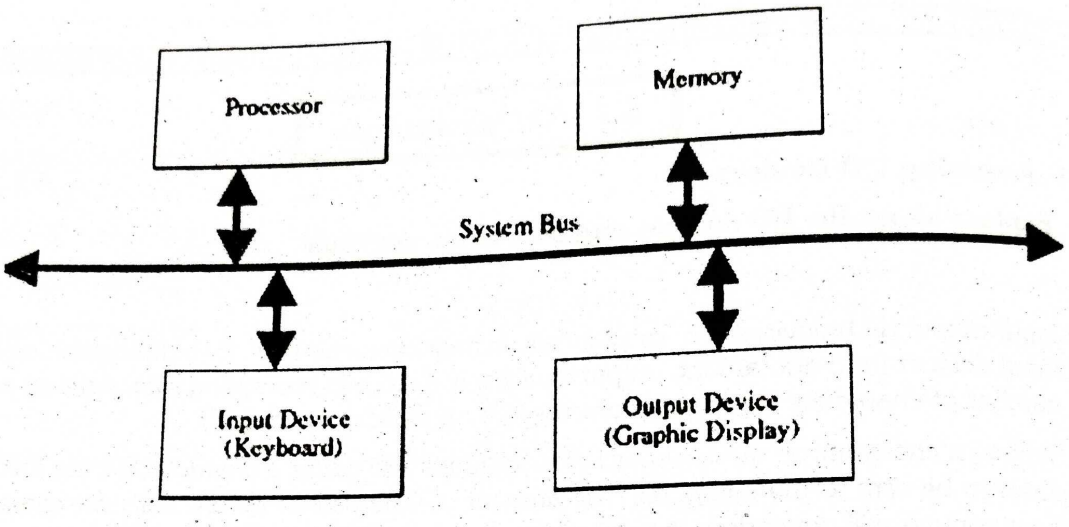


Fig. (a): A single bus system

- Each belonging to a specific output device. In addition, a mechanism according to which the processor can address those input and output registers must be adopted. More than one arrangement exists to satisfy the abovementioned requirements. Among these, two particular methods are explained below.

- In the first arrangement, I/O devices are assigned particular addresses, isolated from the address space assigned to the memory. The execution of an input instruction at an input device address will cause the character stored in the input register of that device to be transferred to a specific register in the CPU. Similarly, the execution of an output instruction at an output device address will cause the character stored in a specific register in the CPU to be transferred to the output register of that output device. This arrangement, called shared I/O, is shown schematically in Figure (b). In this case, the address and data lines from the CPU can be shared between the memory and the I/O devices. A separate control line will have to be used. This is because of the need for executing input and output instructions. In a typical computer system, there exists more than one input and more than one output device. Therefore, there is a need to have address decoder circuitry for device identification. There is also a need for status registers for each input and output device. The status of an input device, whether it is ready to send data to the processor, should be stored in the status register of that device. Similarly, the status of an output device, whether it is ready to receive data from the processor, should be stored in the status register of that device. Input (output) registers, status registers, and address decoder circuitry represent the main components of an I/O interface (module).

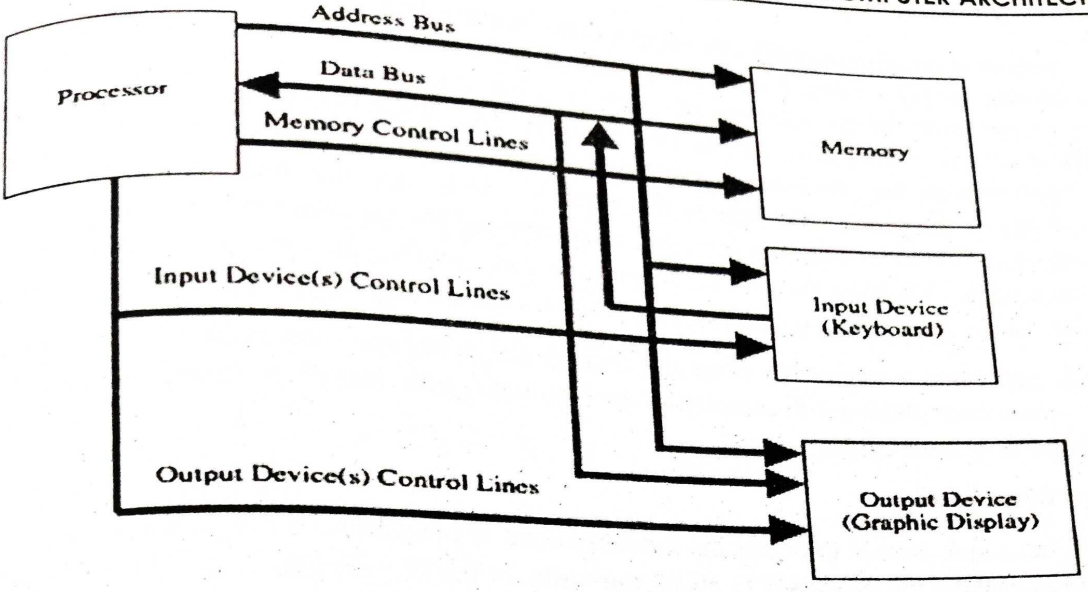


Fig. (b): Shared I/O arrangement

The main advantage of the shared I/O arrangement is the separation between the memory address space and that of the I/O devices. Its main disadvantage is the need to have special input and output instructions in the processor instruction set. The shared I/O arrangement is mostly adopted by Intel.

The second possible I/O arrangement is to deal with input and output registers as if they are regular memory locations. In this case, a read operation from the address corresponding to the input register of an input device, for example, Read Device 6, is equivalent to performing an input operation from the input register in Device #6. Similarly, a write operation to the address corresponding to the output register of an output device, for example, Write Device 9, is equivalent to performing an output operation into the output register in Device #9. This arrangement is called memory-mapped I/O.

The main advantage of the memory-mapped I/O is the use of the read and write instructions of the processor to perform the input and output operations, respectively. It eliminates the need for introducing special I/O instructions. The main disadvantage of the memory-mapped I/O is the need to reserve a certain part of the memory address space for addressing I/O devices, that is, a reduction in the available memory address space. The memory-mapped I/O has been mostly adopted by Motorola.

Q2. Explain three modes of data transfer between I/O devices and a computer system.

Ans :

The three modes of data transfer or three possible techniques of I/O operations are as follows.

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

Programmed I/O

It is a technique of organizing an I/O activity such that the process gives full access to a seperate unit called I/O model to take charge of the entire I/O activity, which (I/O model) inturn does not take any responsibility to intimate the processor about completion of it's task.

In case of programmed I/O whenever a processor indulges itself in the program execution on, (say) encountering an I/O activity, it calls the described I/O module and authorize it. The I/O module after taking charge executes the required I/O activity depending on the availability of the processor. After completion of a task, the I/O module sets the contents of I/O status register and switches to other activities. It neither intimates the processor nor interrupts it to provide the status of execution, rather it is the responsibility of the processor to check the status of the I/O module periodically. Also, if there is any requirement of data to be collected from the main memory, the processor collects it and provides it to the given I/O module. Also here the processor performs various other operations involving checking of the I/O, device, Status, transferring data as well as initiating READ/WRITE command etc.

The processor is ineffective since the time of the processor wasted in handling various meagre activities rather than utilizing it in completing the high value task. Hence, the usage of program driven I/O has declined to greater extent.

Interrupt Driven I/O

This technique is used to overcome the limitations of programmed I/O. In short, in interrupt driven I/O instead of making the processor to check the status of the I/O module, it is the responsibility of the I/O module to intimate the processor by issuing an interrupt signal. The possible cases are as follows.

Case 1

Here the interrupt driven I/O process is analyzed from the I/O module view i.e the I/O module receives a READ signal from the processor. After receiving the signal, the I/O module gets activated and receives the data from the corresponding I/O unit meanwhile, the processor resumes to some other task. As the I/O module completes its task, it intimates the processor about it's status by sending an interrupt through control signals and wait until it receives the response from the processor. The processor responds back to the I/O module by sending the address of the memory(where the required data has to be stored). The I/O module places it's data to the lines and reverts back to the same address.

Case 2

Here the interrupt driven I/O is analyzed from the processors view. Whenever the processor indulges itself in executing the given program and encounter an I/O activity it temporarily sustain it's activity (by placing it's status in the PC and stack pointer), transfer a read command to the I/O module and resumes back to its execution by popping the data from PC and SP registers. The I/O module on receiving command collects the required data from the I/O devices, interrupts the processor, and the entire activity goes on as it was observed in case 1.

Direct memory access (DMA)

This technique is more efficient than the programmed I/O and interrupt driven techniques. DMA or Direct Memory Access is granted to an independent block which resides either on the system bus or near the I/O module. Whenever the processor analyzes the requirement of performing an I/O activity, it activates the given DMA module by transmitting special information consisting of data such as, the address of the given I/O devices, status of READ/WRITE operations, address of the memory location, amount of data to be READ/WRITE etc. and the processor resume back to its execution. The DMA module based on this information performs the given task and interrupts the processor about it's status. Hence, in this process, the processor is involved only at the beginning and at the end. As the I/O devices can now transmit their data or interact with the memory directly, the process is called DMA.

5.1.2 Interrupts

Q3. Explain Programmed I/O with example.

Ans : **(Imp.)**

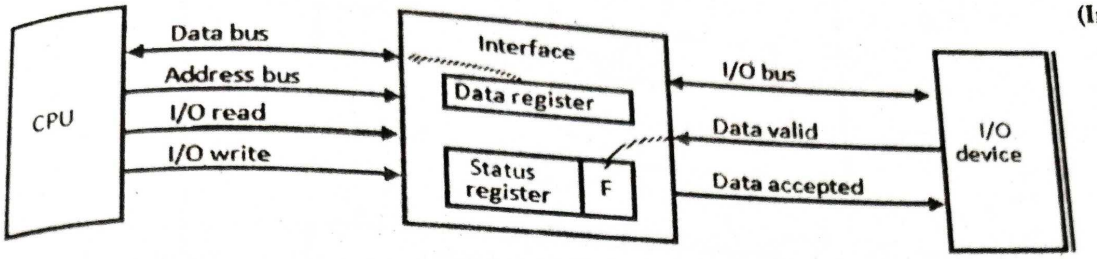


Fig. (a): Data transfer from I/O device to CPU

- In the programmed I/O method, the I/O device does not have direct access to memory.
- An example of data transfer from an I/O device through an interface into the CPU is shown in figure (a).
- When a byte of data is available, the device places it in the I/O bus and enables its data valid line.
- The interface accepts the byte into its data register and enables the data accepted line.
- The interface sets a bit in the status register that we will refer to as an F or "flag" bit.
- The device can now disables the data valid line, but it will not transfer another byte until the data accepted line is disables by the interface.
- A program is written for the computer to check the flag in the status register to determine if a byte has been placed in the data register by the I/O device.
- This is done by reading the status register into a CPU register and checking the value of the flag bit.
- Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.

Example of Programmed I/O

- A flowchart of the program that must be written for the CPU is shown in figure (b).
- It is assumed that the device is sending a sequence of bytes that must be stored in memory.
- The transfer of each byte requires three instructions :
- Read the status register.
- Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.
- Read the data register.
- Each byte is read into a CPU register and then transferred to memory with a store instruction.
- A common I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer.

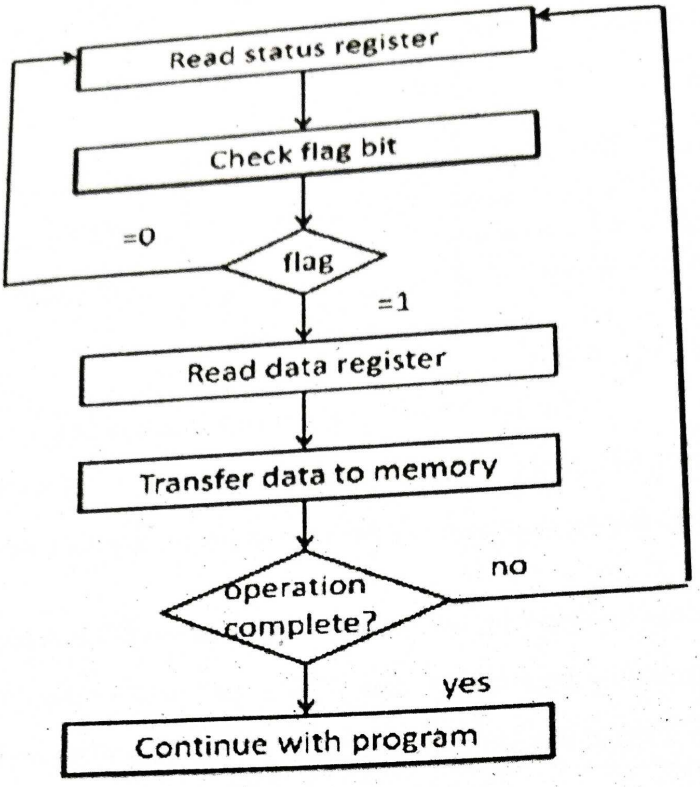


Fig. (b): Flowchart for CPU program to input data

Q4. Write a note on Interrupt Initiated I/O.

Ans :

- In programmed initiated, CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.
- This is a time consuming process since it keeps the processor busy needlessly.
- It can be avoided by using an interrupt facility and a special command to inform the interface to issue an interrupt request signal when data are available from the device.
- In the meantime CPU can proceed to execute another program.
- The interface meanwhile keeps monitoring the device.
- When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.
- While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- The CPU deviates from what it is doing to take care of the input or output transfer.
- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

- The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.
- The way that the processor chooses the branch address of the service routine varies from one unit to another.
- In non-vectored interrupt, branch address is assigned to a fixed location in memory.
- In a vectored interrupt, the source that interrupts supplies the branch information to the computer. The information is called vector interrupt.
- In some computers the interrupt vector is the first address of the I/O service routine.
- In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the I/O service routine is stored.

5.1.3 Priority Interrupt

Q5. What is priority interrupt? Explain Daisy Chaining. **(Imp.)**

Ans. :

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt.

Daisy Chaining Priority

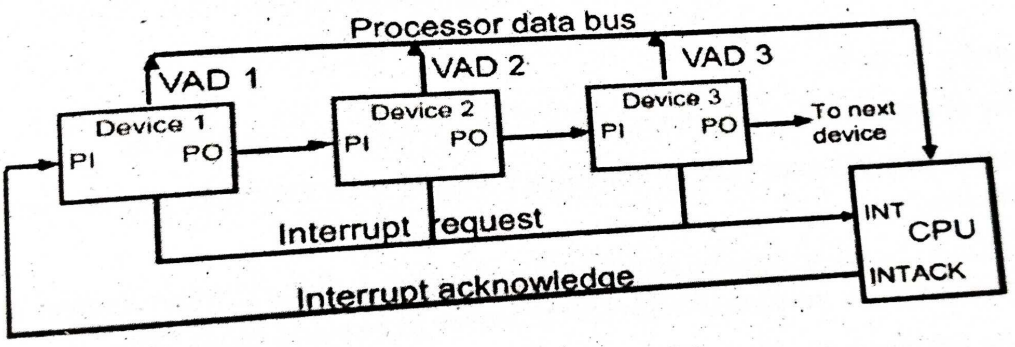


Fig. Daisy-chain priority interrupt

- The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.
- The device with the highest priority is placed in the first position, followed by lower priority devices up to the device with the lowest priority, which is placed last in the chain.
- This method of connection between three devices and the CPU is shown in figure.
- If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU.
- When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU.

- The CPU responds to an interrupt request by enabling the interrupt acknowledge line.
- This signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.
- If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output.
- It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.
- A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower priority device that the acknowledge signal has been blocked.
- A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output.
- If the device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output.
- Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus.
- The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU.
- The farther the device is from the first position; the lower is its priority.

5.2 Direct Memory Access

Q6. Write a detailed note on Direct Memory Access (DMA). **(Imp.)**

Ans :

Direct Memory Access

- Transfer of data under programmed I/O is between CPU and peripheral.
- In direct memory access (DMA), Interface transfers data into and out of memory through the memory bus.
- The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks.
- When the transfer is made, the DMA requests memory cycles through the memory bus.
- When the request is granted by the memory controller, DMA transfers the data directly into memory.

DMA Controller

- DMA controller - Interface which allows I/O transfer directly between Memory and Device, freeing CPU for other tasks.
- CPU initializes DMA Controller by sending memory address and the block size (number of words).

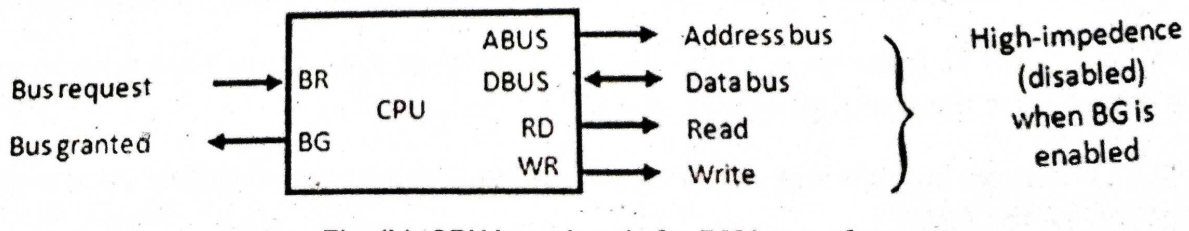


Fig. (b): CPU bus signals for DMA transfer

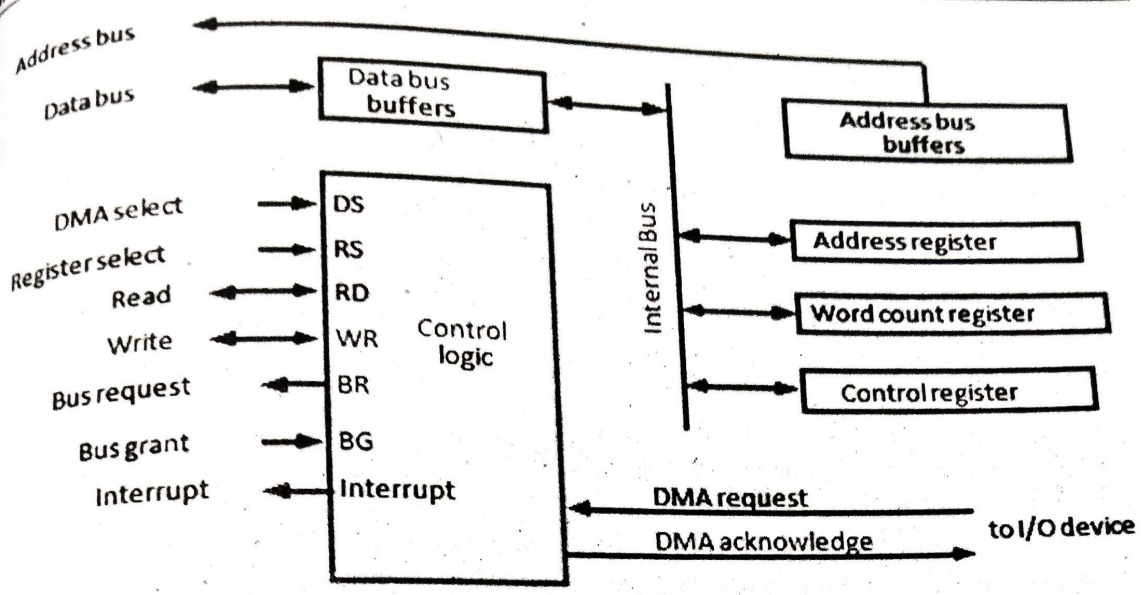


Fig. (b): Block diagram of DMA controller

The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.

In addition, it needs an address register, a word count register, and a set of address lines.

The address register and address lines are used for direct communication with the memory.

The word count register specifies the number of words that must be transferred.

The data transfer may be done directly between the device and memory under control of the DMA.

Figure (b) shows the block diagram of a typical DMA controller.

The unit communicates with the CPU via the data bus and control lines.

The register in the DMA are selected by the CPU through the address bus by enabling the

DS (DMA select) and RS (register select) inputs.

The RD (read) and WR (write) inputs are bidirectional.

When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers.

When BG= 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address but and activating the RD or WR control.

The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.

The DMA controller has three registers: an address register, a word count register, and a control register.

The address register contains an address to specify the desired location in memory.

The word count register holds the number of words to be transferred.

- This register is decremented by one after each word transfer and internally tested for zero.
- The control register specifies the mode of transfer.
- All registers in the DMA appear to the CPU as I/O interface registers.
- Thus the CPU can read from or write into the DMA register under program control via the data bus.
- The DMA is first initialized by the CPU.
- After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred.
- The CPU initializes the DMA by sending the following information through the data bus
- The staring address of the memory block where data are available (for read) or where data are to be stored (for write)
- The word count, which is the number of words in the memory block.
- Control to specify the mode of transfer such as read or write.
- The starting address is stored in the address register.

5.2.1 Buses

Q7. What is computer bus? Write about it.

Ans : **(Imp.)**

A bus in computer terminology represents a physical connection used to carry a signal from one point to another. The signal carried by a bus may represent address, data, control signal, or power. Typically, a bus consists of a number of connections running together. Each connection is called a bus line. A bus line is normally identified by a number. Related groups of bus lines are usually identified by a name. For example, the group of bus lines 1 to 16 in a given computer system may be used to carry the address of memory locations, and therefore are identified as address lines.

Depending on the signal carried, there exist at least four types of buses: address, data, control, and power buses. Data buses carry data, control buses carry control signals, and power buses carry the power-supply/ground voltage. A bus can be synchronous if data transfer over the bus is controlled by a bus clock. The clock acts as the timing reference for all bus signals. A bus is asynchronous if data transfer over the bus is based on the availability of the data and not on a clock signal. Data is transferred over an asynchronous bus using a technique called handshaking. The operations of synchronous and asynchronous buses are explained below.

To understand the difference between synchronous and asynchronous, let us consider the case when a master such as a CPU or DMA is the source of data to be transferred to a slave such as an I/O device. The following is a sequence of events involving the master and slave:

1. **Master:** send request to use the bus
2. **Master:** request is granted and bus is allocated to master
3. **Master:** place address/data on bus
4. **Slave:** slave is selected
5. **Master:** signal data transfer
6. **Slave:** take data
7. **Master:** free the bus

Q8. Explain about synchronous and asynchronous bus protocols.

Ans :

Bus Protocols

A bus is a communication channel shared by many devices and hence rules need to be established in order for the communication to happen correctly. These rules are called bus protocols. Design of a bus architecture involves several tradeoffs related to the width of the data bus, data transfer size, bus protocols, clocking, etc. Depending on whether the bus transactions are controlled by a clock or not, buses are classified into synchronous and asynchronous buses. Depending on whether the data bits are sent on parallel wires or multiplexed onto one single wire, there are parallel and serial buses. Control of the bus communication in the presence of multiple devices necessitates defined procedures called arbitration schemes.

Synchronous Buses

In synchronous buses, the steps of data transfer take place at fixed clock cycles. Everything is synchronized to bus clock and clock signals are made available to both master and slave. The bus clock is a square wave signal. A cycle starts at one rising edge of the clock and ends at the next rising edge which is the beginning of the next cycle. A transfer may take multiple bus cycles depending on the speed parameters of the bus and the two ends of the transfer.One scenario would be that on the first clock cycle, the master puts an address on the address bus, puts data on the data bus, and asserts the appropriate control lines. Slave recognizes its address on the address bus on the first cycle and reads the new value from the bus in the second cycle. Synchronous buses are simple and easily implemented. However, when connecting devices with varying speeds to a synchronous bus, the slowest device will determine the speed of the bus. Also, the synchronous bus length could be limited to avoid clock-skewing problems.

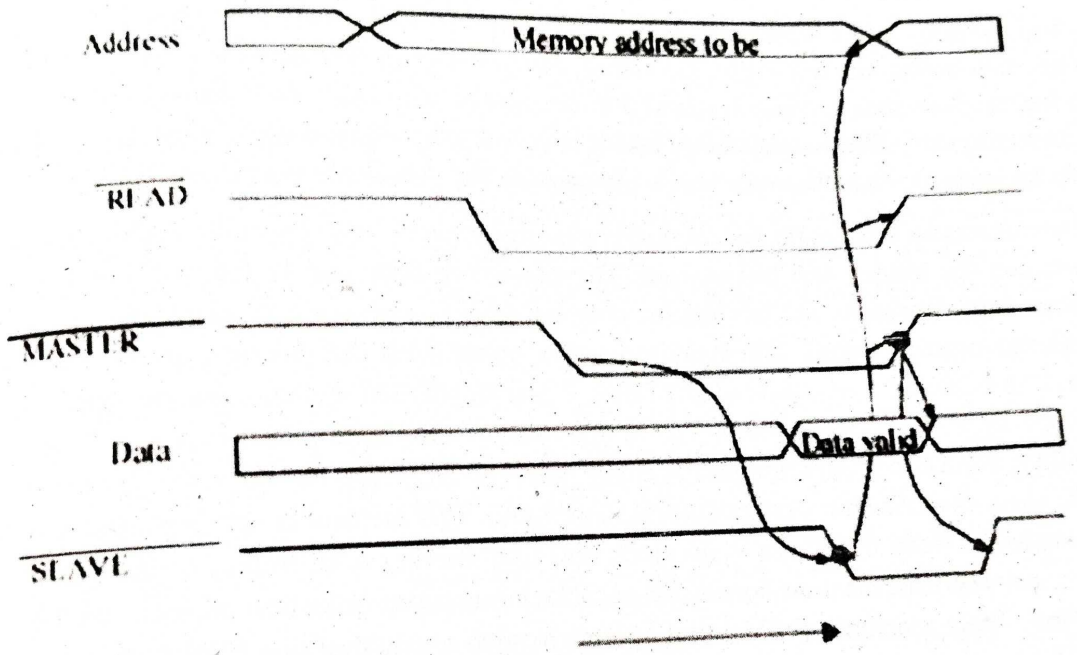


Fig. (a): Read Operations on an Asynchronous Bus

A memory read transaction on the synchronous bus typically proceeds as illustrated in Fig. (a) During the first clock cycle the CPU places the address of the location it wants to read, on the address lines of the bus. Later during the same clock cycle, once the address lines have stabilized, the READ request is asserted by the CPU. Many times, some of these control signals are active low and asserting the signal means that they are pulled low. A few clock cycles are needed for the memory to perform accessing of the requested location. In a simple non-pipelined bus, these appear as wait states and the data is placed on the bus by the memory after the tow or three wait cycles. The CPU then releases the bus by deasserting the READ control signal. The write transaction is similar except that the processor is the data source and the WRITE signal is the one that is asserted. Different bus architectures synchronize bus operations with respect to the rising edge or falling edge or level of the clock signal.

Asynchronous Buses

There are no fixed clock cycles in asynchronous buses. Handshaking is used instead. Figure (b) shows the handshaking protocol. The master asserts the data-ready line until it sees a data-accept signal.

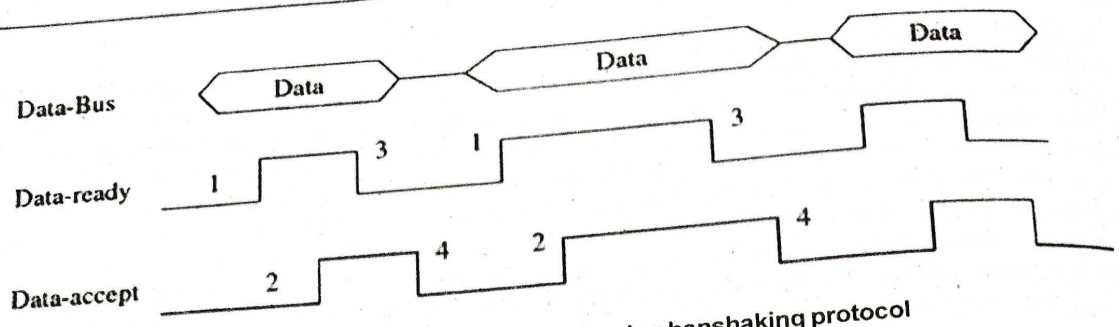


Fig. (b): Asynchronous bus timing using hanshaking protocol

When the slave sees a dataready signal, it will assert the data-accept line (point 2 in the figure). The rising of the data-accept line will trigger the falling of the data-ready line and the removal of data from the bus. The falling of the data-ready line (point 3 in the figure) will trigger the falling of the data-accept line (point 4 in the figure). This handshaking, which is called fully interlocked, is repeated until the data is completely transferred. Asynchronous bus is appropriate for different speed devices.

An asynchronous bus has no system clock. Handshaking is done to properly conduct the transmission of data between the sender and the receiver. The process is illustrated in Fig. (b). For example, in an asynchronous read operation, the bus master puts the address and control signals on the bus and then asserts a synchronization signal. The synchronization signal from the master prompts the slave to get synchronized and once it has accessed the data, it asserts its own synchronization signal. The slave's synchronization signal indicates to the processor that there is valid data on the bus, and it reads the data. The master then deasserts its synchronization signal, which indicates to the slave that the master has read the data. The slave then deasserts its synchronization signal. This method of synchronization is referred to as a full handshake. Note that there is no clock and that starting and ending of the data transfer are indicated by special synchronization signals. An asynchronous communication protocol can be considered as a pair of Finite State machines (FSMs) that operate in such a way that one FSM does not proceed until the other FSM has reached a certain state.

Synchronous buses are typically faster than asynchronous buses because there is no overhead to establish a time reference for each transaction. Another reason that helps the synchronous bus to operate fast is that the bus protocol is predetermined and very little logic is involved in implementing the Finite State machine. However, synchronous buses are affected by clock skew and they cannot be very long. But asynchronous buses work well even when they are long because clock skew problems do not affect them. Thus asynchronous buses can handle longer physical distances and higher number of devices. Processor-memory buses are typically synchronous because the devices connected to the bus are fast, are small in number and are located in close proximity. I/O buses are typically asynchronous because many peripherals need only slow data rates and are physically situated far away.

Q9. What is called bus arbitration? Explain.

Ans :

Bus Arbitration

Bus arbitration is needed to resolve conflicts when two or more devices want to become the bus master at the same time. In short, arbitration is the process of selecting the next bus master from among

multiple candidates. Conflicts can be resolved based on fairness or priority in a centralized or distributed mechanisms. Centralized Arbitration In centralized arbitration schemes, a single arbiter is used to select the next master. A simple form of centralized arbitration uses a bus request line, a bus grant line, and a bus busy line. Each of these lines is shared by potential masters, which are daisy-chained in a cascade. Figure (a) shows this simple centralized arbitration scheme. In the figure, each of the potential masters can submit a bus request at any time.

A fixed priority is set among the masters from left to right. When a bus request is received at the central bus arbiter, it issues a bus grant by asserting the bus grant line. When the potential master that is closest to the arbiter (potential master 1) sees the bus grant signal, it checks to see if it had made a bus request. If yes, it takes over the bus and stops propagation of the bus grant signal any further. If it has not made a request, it will simple turn the bus grant signal to the next master to the right (potential master 2), and so on. When the transaction is complete, the busy line is deasserted.

Instead of using shared request and grant lines, multiple bus request and bus grant lines can be used. In one scheme, each master will have its own independent request and grant line as shown in Figure (b). The central arbiter can employ any prioritybased or fairness-based tiebreaker. Another scheme allows the masters to have multiple priority levels. For each priority level, there is a bus request and a bus grant line. Within each priority level, daisy chain is used. In this scheme, each device is attached to the daisy chain of one priority level.

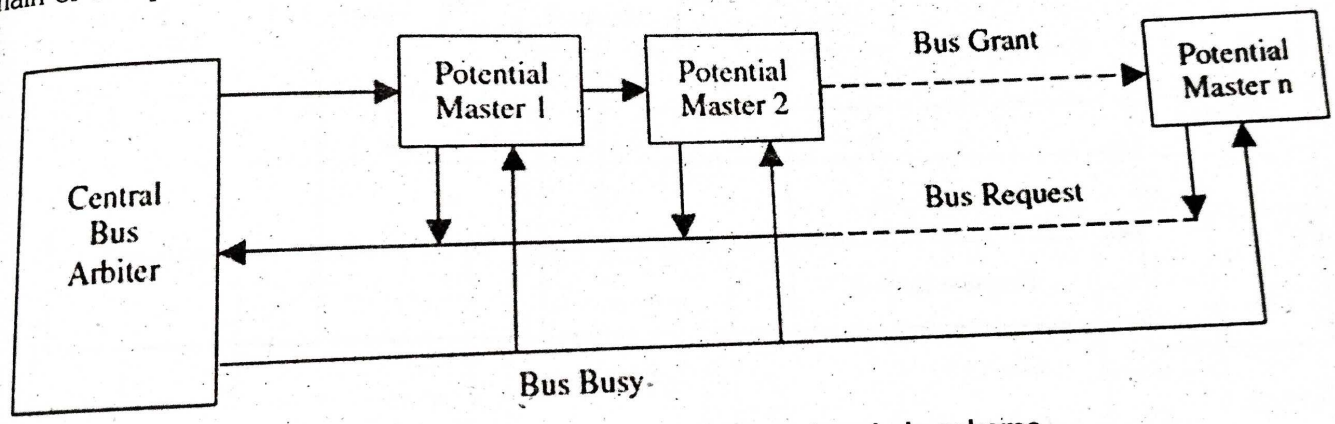


Fig. (a): Centralized arbiter in a daisy-chain scheme

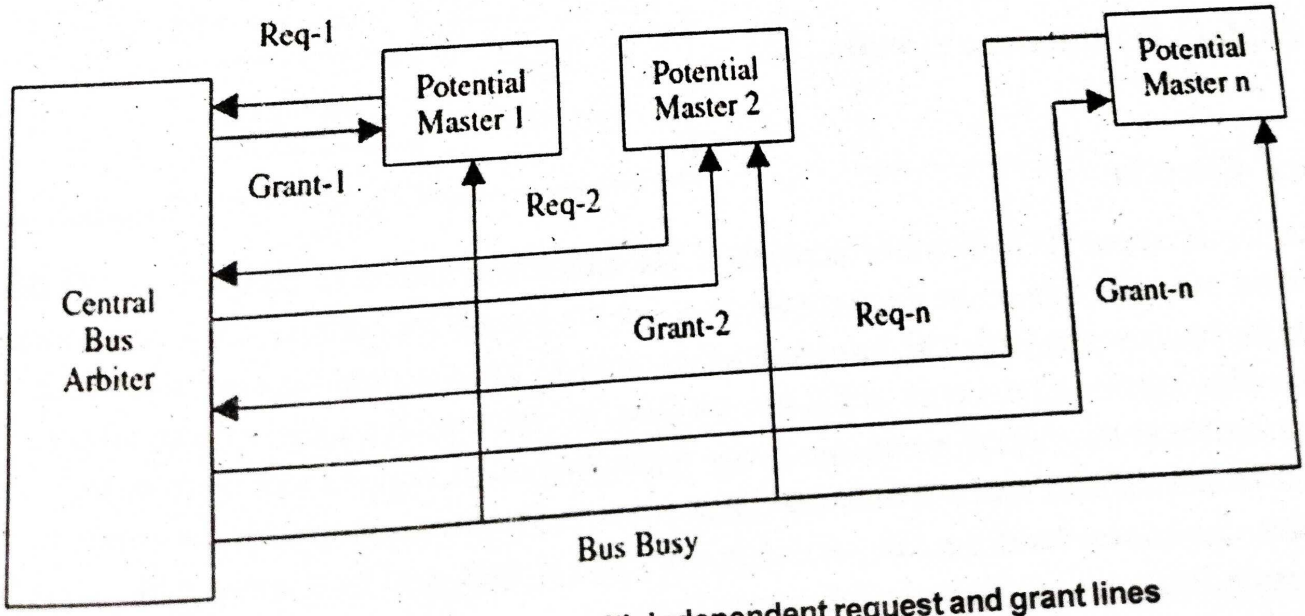


Fig. (b): Centralized arbiter with independent request and grant lines

If the arbiter receives multiple bus requests from different levels, it grants the bus to the level with the highest priority. Daisy chaining is used among the devices of that level. Figure (c) shows an example of four devices included in two priority levels. Potential master 1 and potential master 3 are daisy-chained in level 1 and potential master 2 and potential master 4 are daisy-chained in level 2.

Decentralized Arbitration

In decentralized arbitration schemes, priority-based arbitration is usually used in a distributed fashion. Each potential master has a unique arbitration number, which is used in resolving conflicts when multiple requests are submitted. For example, a conflict can always be resolved in favor of the device with the highest arbitration number. The question now is how to determine which device has the highest arbitration number? One method is that a requesting device would make its unique arbitration number available to all other devices. Each device compares that number with its own arbitration number. The device with the smaller number is always dismissed. Eventually, the requester with the highest arbitration number will survive and be granted bus access.

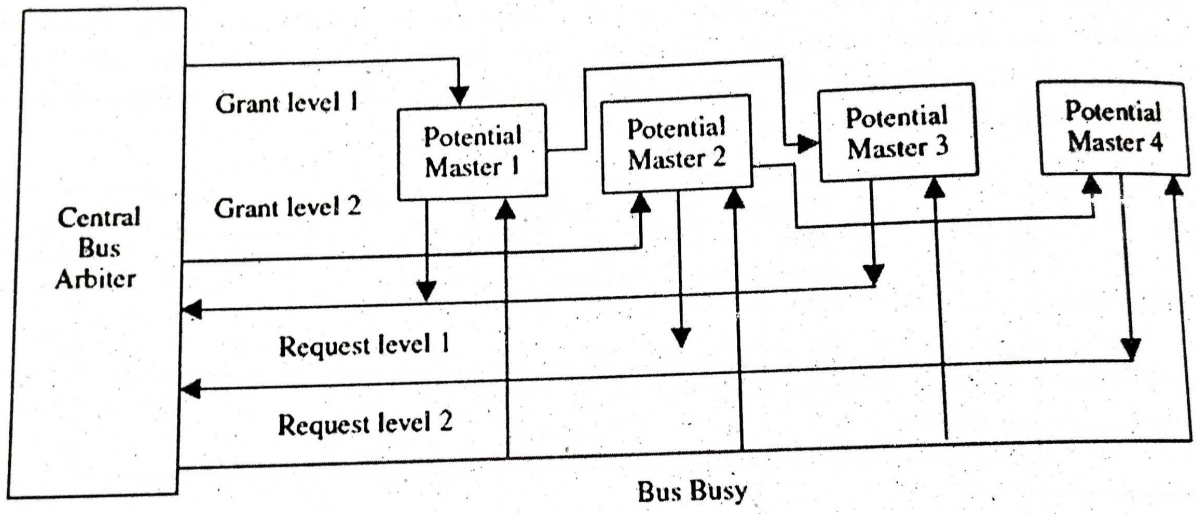


Fig. (c): Centralized arbiter with two priority levels (four devices)

5.2.2 Interface Circuits

Q10. Explain I/O interface circuits. **(Imp.)**

Ans :

Interface Circuits

An Input/output (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 the bus signals for address, data, and control. On the other side we have a data path with its associated controls to transfer data between the interface and the I/O device. This side is called a port, and it can be classified as either a parallel or a serial port. A parallel port transfers data in the form of a number of bits, typically 8 or 16, simultaneously to or from the device. A 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. I/O interface does the following:

1. Provides a storage buffer for at least one word of data (or one byte, in the case of byte-oriented devices).
2. Contains status flags that can be accessed by the processor to determine whether the buffer is full (for input) or empty (for output)
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

Figure 5.20 shows the hardware components needed for connecting a keyboard to a processor. A typical keyboard consists of mechanical switches that are normally open. When a key is pressed, its switch closes and establishes a path for an electrical signal. This signal is detected by an encoder circuit that generates the ASCII code for the corresponding character. A difficulty with such push-button switches is that the contacts bounce when a key is pressed. Although bouncing may last only one or two milliseconds, this is long enough for the computer to observe a single pressing of a key as several distinct electrical events; this single pressing could be erroneously interpreted as the key being pressed and released rapidly several times. The effect of bouncing must be eliminated. We can do this in two ways: A simple de-bouncing circuit can be included, or a software approach can be used. When debouncing is implemented in software, the I/O routine that reads a character from the keyboard waits long enough to ensure that bouncing has subsided. Figure (a) illustrates the hardware approach; debouncing circuits are included as a part of the encoder block.

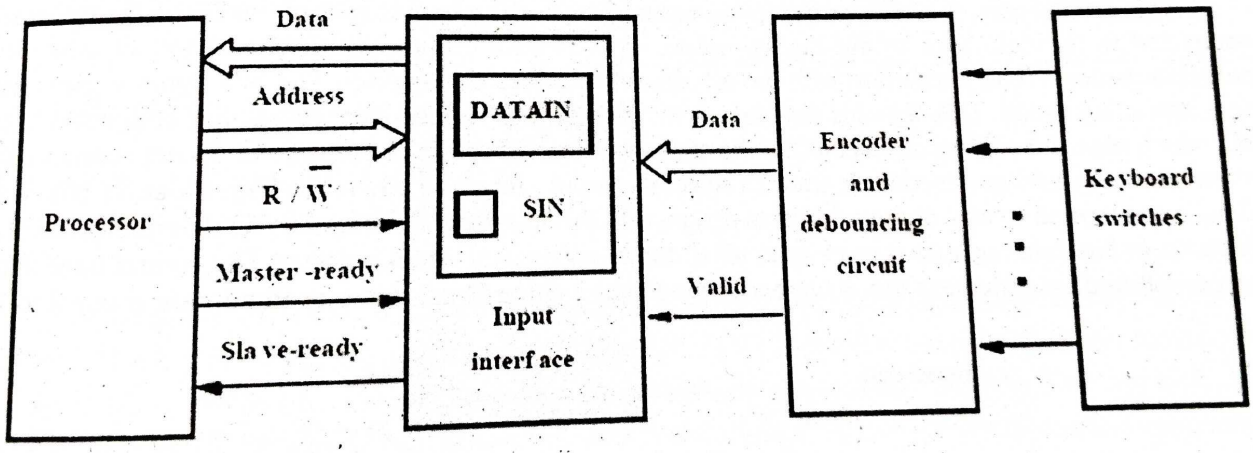


Fig. (a): Keyboard to processor connection

The output of the encoder consists of the bits that represent the encoded character and one control signal called Valid, which indicates that a key is being pressed. This information is sent to the interface circuit, which contains a data register, DATAIN, and a status flag, SIN. When a key is pressed, the valid signal changes from 0 to 1,, causing the ASCII code to be loaded into DATAIN and SIN to be set to 1. The status flag SIN is cleared to 0 when the processor reads the contents of the DATAIN register. The interface circuit is connected to an asynchronous bus on which transfers are controlled using the handshake signals Master-ready and Slave-ready. The third control line, R/W distinguishes read and write transfers.

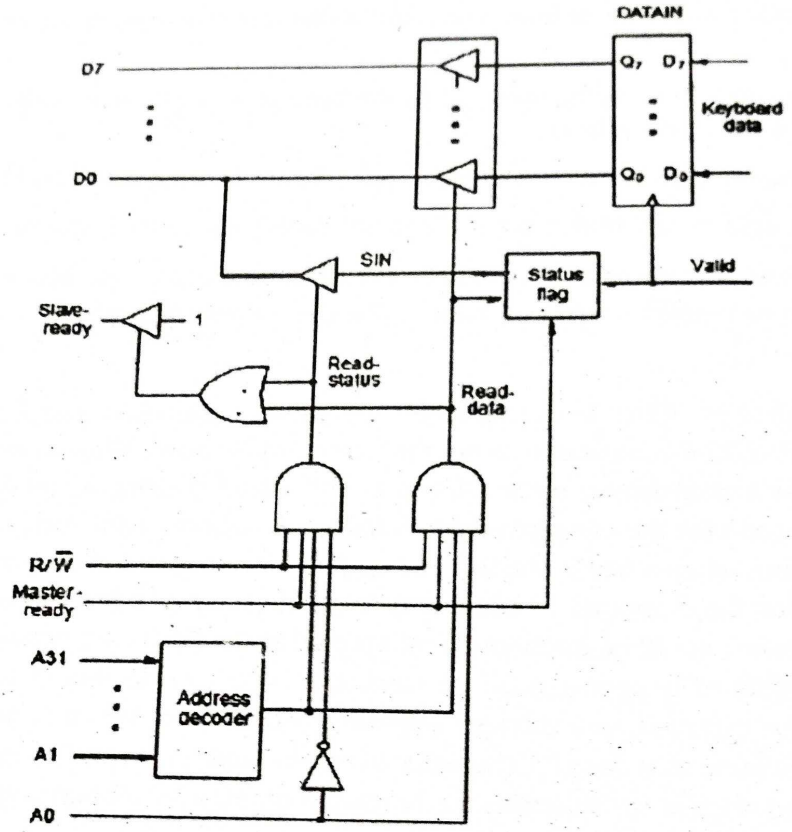


Fig. (b): Circuit for Input Interface

Figure (b) shows a suitable circuit for an input interface. The output lines of the DATAIN register are connected to the data lines of the bus by means of three-state drivers, which are turned on when the processor issues a read instruction with the ad- dress that selects this register. The SIN signal is generated by a status flag circuit. This signal is also sent to the bus through a three-state driver. It is connected to bit DO, which means it will appear as bit 0 of the status register. Other bits of this register do not contain valid information. An address decoder is used to select the input interface when the high-order 31 bits of an address correspond to any of the addresses assigned to this interface. Address bit AO determines whether the status or the data registers is to be read when the Master-ready signal is active. The control handshake is accomplished by activating the Slaveready signal when either Read-status or Read-data is equal to 1.

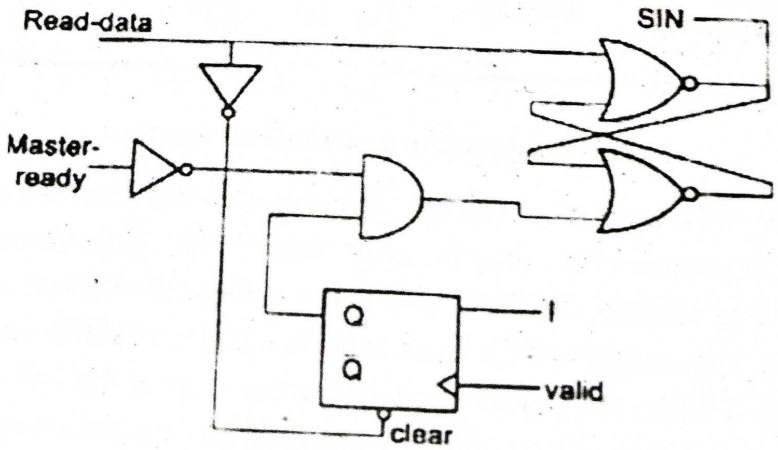


Fig. (c): Circuit for the status Rag block in Figure

A possible implementation of the status flag circuit is shown in Figure (c). An edgetriggered D flip-flop is set to 1 by a rising edge on the Valid signal line. this event changes the state of the NOR latch such that SIN is set to 1. The state of this latch must not change while SIN is being read by the processor. Hence, the circuit ensures that SIN can be set only while Masterready is equal to 0. Both the flip- flop and the latch are reset to 0 when Read-data is set to 1 to read the DATAIN register.

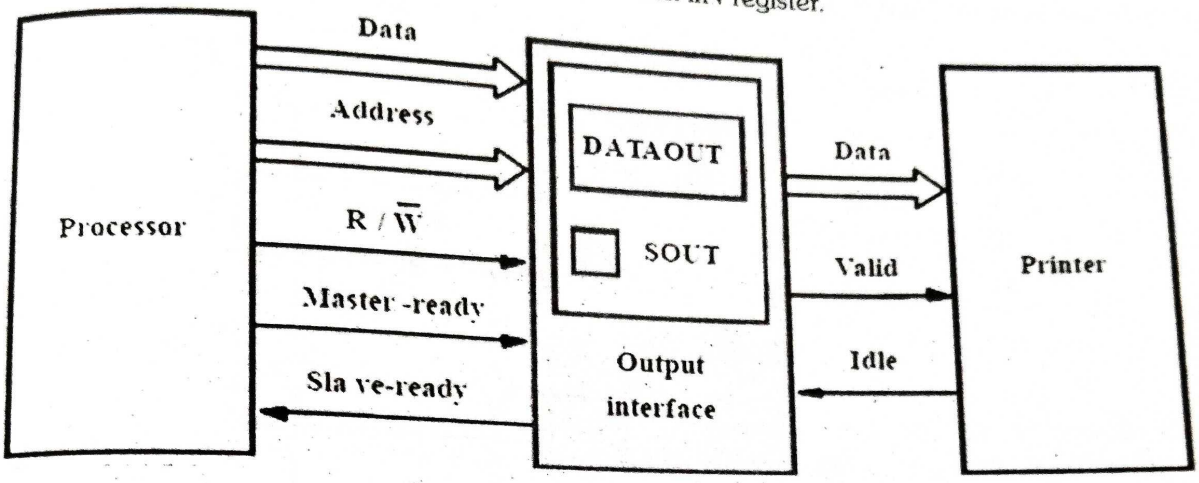


Fig.(d): Printer to processor connection

Let us now consider an output interface that can be used to connect an output device, such as a printer, to a processor, as shown in Figure (d). The printer operates under control of the handshake signals Valid and Idle in a manner similar to the handshake used on the bus with the Master-ready and Slave-ready signals. When it is ready to accept a character, the printer asserts its Idle signal. The interface circuit can then place a new character on the data lines and activate the Valid signal. In response, the printer starts printing the new character and negates the Idle signal, which in turn causes the interface to deactivate the Valid signal.

5.2.3 Standard I/O Interfaces (Pci Scsi Usb)

Q11. Write a note on serial interface.

(Imp.

Ans :

A serial port is used to connect the processor to I/O devices that require transmission of data one b at a time. The key feature of an interface circuit for a serial port is that it is capable of communicating in bit-serial fashion on the device side and in a bit-parallel fashion on the bus side. The transformatic between the parallel and serial formats is achieved with shift registers that have parallel access capability. block diagram of a typical serial interface is shown in Figure. It includes the familiar DATAIN and DATAOU registers. The input shift register accepts bit-serial input from the I/O device. When all 8 bits of data ha been received, the contents of this shift register are loaded in parallel into the DATAIN register. Simila output data in the DATAOUT register are loaded into the output shift register, from which e bits are shif out and sent to the I/O device.

The part of the interface that deals with the bus is the same as in the parallel interface descri earlier. The status flags SIN and SOUT serve similar functions. The SIN flag is set to 1 when new data

loaded in DATAIN; it is cleared to 0 when the processor reads the contents of DATAIN. As soon as the data are transferred from the input shift register into the DATAIN register, the shift register can start accepting the next 8-bit character from the I/O device. The SOUT flag indicates whether the output buffer is available. It is cleared to 0 when the processor writes new data into the DATAOUT register and set to 1 when data are transferred from DATAOUT into the output shift register.

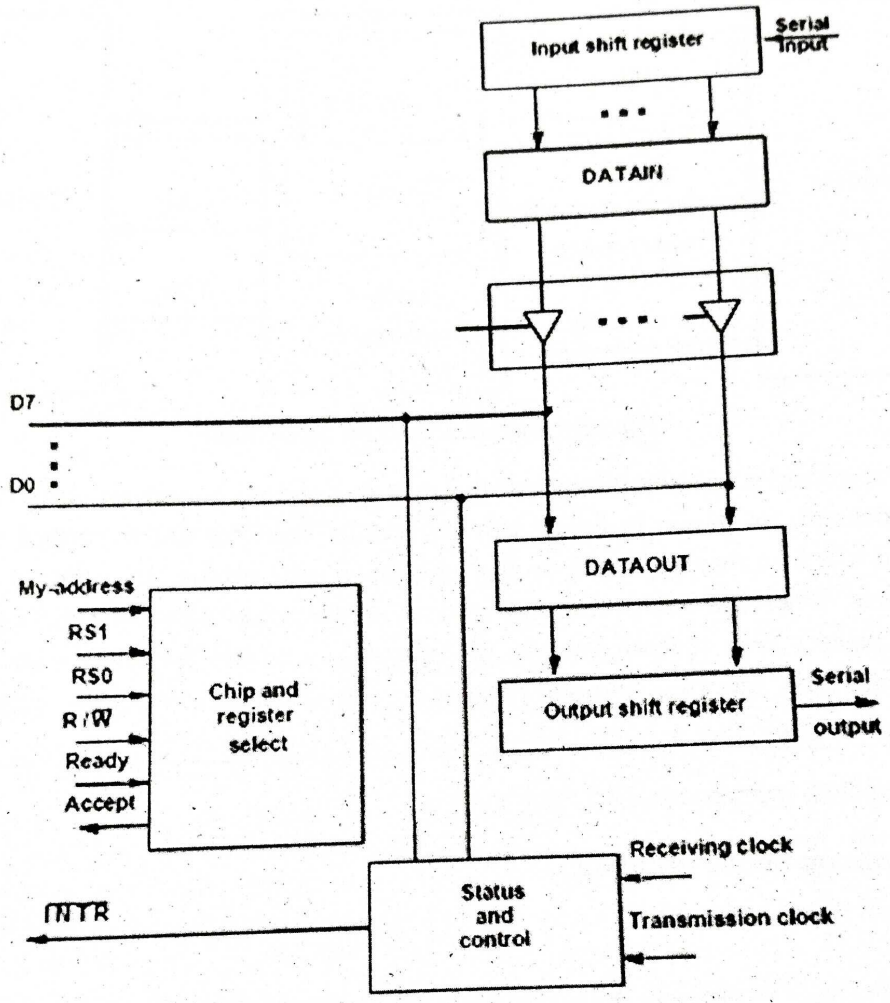


Fig.: A serial interface

The double buffering used in the input and output paths are important. A simpler interface could be implemented by turning DATAIN and DATA OUT into shift registers and eliminating the shift registers in Figure. However, this would impose awkward restrictions on the operation of the I/O device; after receiving one character from the serial line, the device cannot start receiving the next character until the processor reads the contents of DATAIN. Thus, a pause would be needed between two characters to allow the processor to read the input data. With the double buffer, the transfer of the second character can begin as soon as the first character is loaded from the shift register into the DATAIN register. Thus, provided the processor reads the contents of DATAIN before the serial transfer of the second character is completed, the interface can receive a continuous stream of serial data. An analogous situation occurs in the output path of the interface.

Because it requires fewer wires, serial transmission is convenient for connecting devices that are physically far away from the computer. The speed of transmission, often given as a bit rate, depends on the nature of the devices connected. To accommodate a range of devices, a serial interface must be able to use a range of clock speeds. The circuit in Figure allows separate clock signals to be used for input and output operations for increased flexibility. Because serial interfaces play a vital role in connecting I/O devices, several widely used standards have been developed. A standard circuit that includes the features of our example in Figure is known as a Universal Asynchronous Receiver Transmitter (UART). It is intended for use with low-speed serial devices. Data transmission is performed using the asynchronous start-stop format. To facilitate connection to communication links, a popular standard known as RS-232-C was developed.

Q12. Explain PCI interface. **(Imp.)**

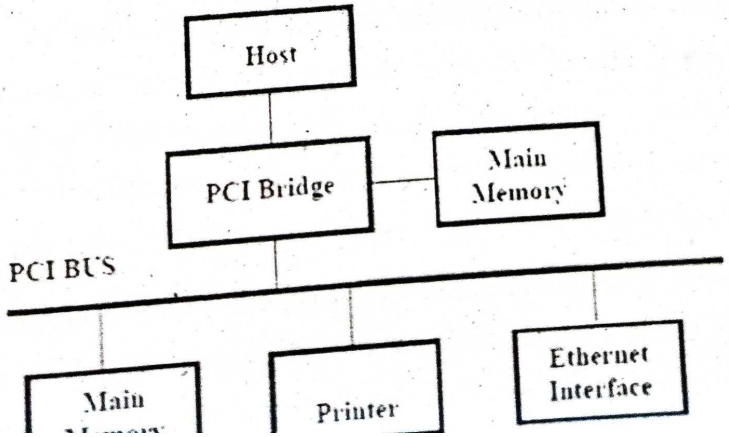
Ans :

PCI

The PCI was developed as a low-cost bus that is truly processor independent. Its design anticipated a rapidly growing demand for bus bandwidth to support high-speed disks and graphic and video devices, as well as the specialized needs of multiprocessor systems. As a result, the PCI is still popular as an industry standard almost a decade after it was first introduced in 1992. An important feature that the PCI pioneered is a plug-and-play capability for connecting I/O devices. To connect a new device, the user simply connects the device interface board to the bus. The software takes care of the rest.

Data Transfer

Most memory transfers involve a burst of data rather than just one word. The reason is that modem processors include a cache memory. The PCI is designed primarily to support this mode of operation. A read or a write operation involving a single word is simply treated as a burst of length one. The bus supports three independent address spaces: memory, I/O, and configuration. The first two are self-explanatory. The I/O address space is intended for use with processors, such as Pentium, that have a separate I/O address space. Figure shows the main memory of the computer connected directly to the processor bus. An alternative arrangement that is used often with the PCI bus is shown in Figure. The PCI Bridge provides a separate physical connection for the main memory. For electrical reasons, the bus may be further divided into segments connected via bridges. However, regardless of which bus segment a device is connected to, it may still be mapped into the processor's memory address space.



At any given time, one device is the bus master. It has the right to initiate data transfers by issuing read and write commands. A master is called an initiator in PCI terminology. This is either a processor or a DMA controller. The addressed device that responds to read and write commands is called a target. The main bus signals used for transferring data are listed in Table. Signals whose name ends with the symbol # are asserted when in the low- voltage state. The main difference between the PCI protocol with others is that in addition to a Target-ready signal, PCI also uses an Initiatorready signal, IRDY #. The latter is needed to support burst transfers.

Table: Data transfer si9na on the PCI bus.

Name	Function
CLK	A 33-MHz or 66-MHz clock.
FRAMED	Sent by the initiator to indicate the duration of a transaction.
AD	32 address/data lines, which may be OJtionally 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.

Consider a bus transaction in which the processor reads four 32-bit words from the memory. In this case, the initiator is the processor and the target is the memory. A complete transfer operation on the bus, involving an address and a burst of data, is called a transaction. Individual word transfers within a transaction are called phases. A clock signal provides the timing reference used to coordinate different phases of a transaction. All signal transitions are triggered by the rising edge of the clock. The signals changing later in the clock cycle to indicate the delays they encounter.

Q13. Explain SCSI (Small Computer System Interface).

Ans :

SCSI (Small Computer System Interface)

The acronym SCSI stands for Small Computer System Interface. It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X3.131. In the original specifications of the standard, devices such as disks are connected to a computer via a 50- wire cable, which can be up to 25 meters in length and can transfer data at rates up to 5 megabytes/s. The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years. SCSI-2 and SCSI-3 have been defined, and each has several options.

A SCSI bus may have eight data lines, in which case it is called a narrow bus and transfers data one byte at a time. Because of these various options, the SCSI connector may have 50, 68, or 80 pins. The maximum transfer rate in commercial devices that are currently available varies from 5 megabytes/s to 160 megabytes/so The most recent version of the standard is intended to support transfer rates up to 320 megabytes/s, and 640 megabytes/s is anticipated a little later.

Devices connected to the SCSI bus are not part of the address space of the processor in the same way as devices connected to the processor bus. The SCSI bus is connected to the processor bus through a SCSI controller. This controller uses DMA to transfer data packets from the main memory to the device, or vice versa. A packet may contain a block of data, commands from the processor to the device, or status information about the device.

A controller connected to a SCSI bus is one of two types - an initiator or a target. An initiator has the ability to select a particular target and to send commands specifying the operations to be performed. The disk controller operates as a target. It carries out the commands it receives from the initiator. The initiator establishes a logical connection with the intended target. Once this connection has been established, it can be suspended and restored as needed to transfer commands and bursts of data. While a particular connection is suspended, other devices can use the bus to transfer information. This ability to overlap data transfer requests is one of the key features of the SCSI bus that leads to its high performance.

Data transfers on the SCSI bus are always controlled by the target controller. To send a command to a target, an initiator requests control of the bus and, after winning arbitration, selects the controller it wants to communicate with and hands control of the bus over to it. Then the controller starts a data transfer operation to receive a command from the initiator. Assume that the processor wishes to read a block of data from a disk drive and that these data are stored in two disk sectors that are not contiguous. The processor sends a command to the SCSI controller, which causes the following sequence of events to take place:

1. The SCSI controller, acting as an initiator, contends for control of the bus.
2. When the 'initiator wins the arbitration process, it selects the target controller and hands over control of the bus to it.
3. The target starts an output operation (from initiator to target); in response to this, the initiator sends a command specifying the required read operation.
4. The target, realizing that it first needs to perform a disk seek operation, sends a message to the initiator indicating that it will temporarily suspend the connection between them. Then it 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. Then, it reads the data stored in that sector and stores them in a data buffer. When it is ready to begin transferring data to the initiator, the target requests control of the bus. After it wins arbitration, it res elects the initiator controller, thus restoring the suspended connection.
6. The target transfers the contents of the data buffer to the initiator and then suspends the connection again. Data are transferred either 8 or 16 bits in parallel, depending on the width of the bus.
7. The target controller sends a command to the disk drive to perform another seek operation. Then, it transfers the contents of the second disk sector to the initiator, as before. At the end of this transfer, the logical connection between the two controllers is terminated.
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 to inform it that the requested operation has been completed.

This scenario shows that the messages exchanged over the SCSI bus are at a higher level than those exchanged over the processor bus. The SCSI bus standard defines a wide range of control messages that can be ex- changed between the controllers to handle different types of I/O devices. Messages are also defined to deal with various error or failure conditions that might arise during device operation or data transfer.

Q14. Explain USB (Universal Serial Bus).

Ans :

USB (Universal Serial Bus)

Universal Serial Bus (USB) is an industry standard developed through a collaborative effort

of several computer and communications companies, including Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, Nortel Networks, and Philips. USB is a simple and low cost mechanism to connect the devices such as keyboards, mouse, cameras, speakers, printer and display devices to the computer.

The USB supports two speeds of operation, called low-speed (1.5 megabits/s) and fullspeed (12 megabits/s). The most recent revision of the bus specification (USB 2.0) introduced a third speed of operation, called high-speed (480 megabits/s). The USB is quickly gaining acceptance in the market place, and with the addition of the high-speed capability it may well become the interconnection method of choice for most computer devices. The USB has been designed to meet several key objectives:

- Provide a simple, low-cost, and easy to use interconnection system that overcomes the difficulties due to the limited number of I/O ports available on a computer
- Accommodate a wide range of data transfer characteristics for I/O devices, including telephone and Internet connections
- Enhance user convenience through a "plug-and-play" mode of operation

Port limitation

Only a few ports are provided in a typical computer. To add new ports, a user must open the computer box to gain access to the internal expansion bus and install a new interface card. The user may also need to know how to configure the device and the software. An objective of the USB is to make it possible to add many devices to a computer system at any time, without opening the computer box.

Device Characteristics

The different kinds of devices may be connected to a computer cover a wide range of functionality. The speed, volume, and timing constraints associated with data transfers to and from such devices vary significantly. In the case of a keyboard, one byte of data is generated every time a key is pressed, which may happen at any time. These data should be transferred to the computer promptly. Since the event of pressing a key is not synchronized to any other event in -8 computer system, the data generated by the keyboard are called asynchronous. Furthermore, the rate at which the data are generated is quite low. It is limited by the speed of the human operator to about 100 bytes per second, which is less than 1000 bits per second.

Let us consider a different source of data. Many computers have a microphone either externally attached or built in. The sound picked up by the microphone produces an analog electrical signal, which must be converted into a digital form before it can be handled by the computer. This is accomplished by sampling the analog signal periodically. For each sample, an analog-to-digital (A/D) converter generates an n-bit number representing the magnitude of the sample. The number of bits, n, is selected based on, the desired precision with which to represent each sample. Later, when these data are sent to a speaker, a digital-to-analog (D/A) converter is used to restore the original analog signal from the digital format. The sampling process yields a continuous stream of digitized samples that arrive at regular intervals, synchronized with the sampling clock. Such a data stream is called isochronous, meaning that successive events are separated by equal periods of time.

Plug-and-play

The plug-and-play feature means that a new device, such as an additional speaker, can be connected at any time while the system is operating. The system should detect the existence of this new device automatically, identify the appropriate device-driver soft- ware and any other facilities needed to service that device, and establish the appropriate addresses and logical connections to enable them to communicate. The plug-and-play requirement has many implications at all levels in the system, from the hardware to the operating system and the applications software. One of the primary objectives of the design of the USB has been to provide a plug-and-play capability.

Q15. Explain USB architecture. **(Imp.)**

Ans :

USB Architecture

A serial transmission format has been chosen for the USB because a serial bus satisfies the low-cost and flexibility requirements. Clock and data information are encoded together and transmitted as a single signal. Hence, there are no limitations on clock frequency or distance arising from data skew. Therefore, it is possible to provide a high data transfer bandwidth by using a high clock frequency. As pointed out earlier, the USB offers three bit rates, ranging from 1.5 to 480 megabits/s, to suit the needs of different I/O devices.

To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure shown in Figure (a). Each node of the tree has a device called a hub, which 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, speaker, or digital TV), which are called functions in USB terminology.

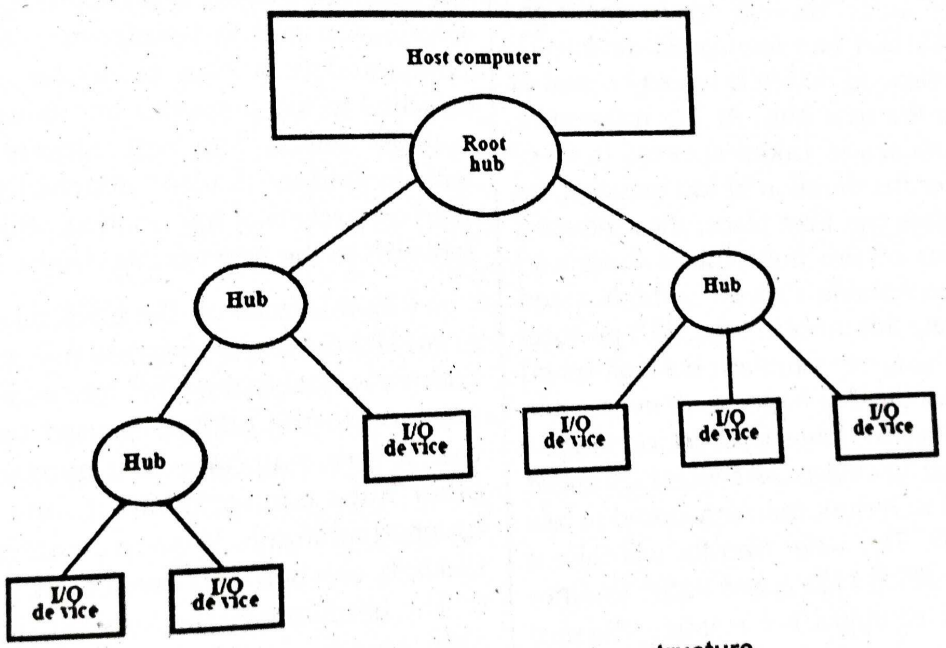


Fig. (a): Universal Serial Bus tree structure.

The tree structure enables many devices to be connected while using only simple point -topoint serial links. Each hub has a number of ports where devices may be connected, including other hubs. In normal operation, 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 ill VO devices, but only the addressed device will respond to that message. A message from an I/O device is sent only upstream towards the root of the tree and is not seen by other devices. Hence, the USB enables the host to communicate with the I/O devices, but it does not enable these devices to communicate with each other.

The USB operates strictly on the basis of polling. A device may send a message only in response to a poll message from the host. Hence, upstream messages do not encounter conflicts or interfere with each other, as no two devices can send messages at the same time. This restriction allows hubs to be simple, Low-cost devices.

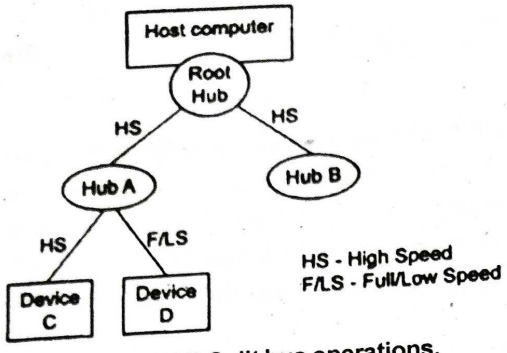


Fig. (b): USB Split bus operations.

The mode of operation described above is observed for all devices operating at either low speed or full speed. However, one exception has been necessitated by the introduction of highpeed operation in USB version 2.0. Consider the situation in Figure (b). Hub A is connected to the root hub by a high-speed link. This hub serves one high-speed device, C, and one low-speed device, D. Normally, a message to device D would be sent at low speed from the root hub. At 1.5 megabits/s, even a short message takes several tens of microseconds. For the duration of this message, no other data transfers can take place, thus reducing the effectiveness of the high-speed links and introducing unacceptable delays for high-speed devices. To mitigate this problem, the USB protocol requires that a message transmitted on a high-speed link is always transmitted at high speed, even when the ultimate receiver is a low-speed device. Hence, a message intended for device D is sent at high speed from the root hub to hub A, then forwarded at low speed to device D. The latter transfer will take a long time, during which high-speed traffic to other nodes is allowed to continue. For example, the root hub may exchange several messages with device C while the low-speed message is being sent from hub A to device D. During this period, the bus is said to be split between high-speed and low-speed traffic. The message to device D is preceded and followed by special commands to hub A to start and end the split-traffic mode of operation, respectively.

The USB standard specifies the hardware details of USB interconnections as well as the organization and requirements of the host software. The purpose of the USB software is to provide bidirectional communication links between application software and I/O devices. These links are called pipes. Any data entering at one end of a pipe is delivered at the other end. Issues such as addressing, timing, or error detection and recovery are handled by the USB protocols. The software that transfers data to or from a given IJO device is called the device driver for that device. The device drivers depend on the characteristics of the devices they support. Hence, a more precise description of the USB pipe is that it connects an VO device to its device driver. It is established when a device is connected and assigned a unique address by the USB software. Once established, data may flow through the pipe at any time.

Addressing

I/O devices are normally identified by assigning them a unique memory address. In fact, a device usually has several addressable locations to enable the software to send and receive control and status information and to transfer data. When a USB is connected to a host computer, its root hub is attached to the processor bus, where it appears as a single device. The host software communicates with individual devices attached to the USB by sending packets of information, which the root hub forwards to the appropriate device in the USB tree.

Each device on the USB, whether it is a hub or an IJO device, 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 to which this device is connected is capable of detecting that the device 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.

Short Question and Answers

1. I/O devices

Ans :

- Input-output (I/O) devices vary substantially in their characteristics. One distinguishing factor among input devices (and also among output devices) is their data processing rate, defined as the average number of characters that can be processed by a device per second.
- Striking a character on the keyboard of a computer will cause a character (in the form of an ASCII code) to be sent to the computer. The amount of time passed before the next character is sent to the computer will depend on the skill of the user and even sometimes on his/her speed of thinking. It is often the case that the user knows what he/she wants to input, but sometimes they need to think before touching the next button on the keyboard. Therefore, input from a keyboard is slow and burst in nature and it will be a waste of time for the computer to spend its valuable time waiting for input from slow input devices. A mechanism is therefore needed whereby a device will have to interrupt the processor asking for attention whenever it is ready. This is called interrupt-driven communication between the computer and I/O devices .
- Consider the case of a disk. A typical disk should be capable of transferring data at rates exceeding several million bytes/second. It would be a waste of time to transfer data byte by byte or even word by word. Therefore, it is always the case that data is transferred in the form of blocks, that is, entire programs. It is also necessary to provide a mechanism that allows a disk to transfer this huge volume of data without the intervention of the CPU. This will allow the CPU to perform other useful operation(s) while a huge amount of data is being transferred between the disk and the memory.

2. Programmed I/O

Ans :

It is a technique of organizing an I/O activity such that the process gives full access to a seperate unit called I/O model to take charge of the entire I/O activity, which (I/O model) inturn does not take any responsibility to intimate the processor about completion of it's task.

In case of programmed I/O whenever a processor indulges itself in the program execution on, (say) encountering an I/O activity, it calls the described I/O module and authorize it. The I/O module after taking charge executes the required I/O activity depending on the availability of the processor. After completion of a task, the I/O module sets the contents of I/O status register and switches to other activities. It neither intimates the processor nor interrupts it to provide the status of execution, rather it is the responsibility of the processor to check the status of the I/O module periodically. Also, if there is any requirement of data to be collected from the main memory, the processor collects it and provides it to the given I/O module. Also here the processor performs various other operations involving checking of the I/O, device, Status, transferring data as well as initiating READ/WRITE command etc.

The processor is ineffective since the time of the processor wasted in handling various meagre activities rather than utilizing it in completing the high value task. Hence, the usage of program driven I/O has declined to greater extent.

3. Direct memory access (DMA)

Ans :

This technique is more efficient than the programmed I/O and interrupt driven techniques. DMA or Direct Memory Access is granted to an independent block which resides either on the system bus or near the I/O module. Whenever the processor analyzes the requirement of performing an I/O activity, it activates the given DMA module

by transmitting special information consisting of data such as, the address of the given I/O devices, status of READ/WRITE operations, address of the memory location, amount of data to be READ/WRITE etc. and the processor resume back to its execution. The DMA module based on this information performs the given task and interrupts the processor about it's status. Hence, in this process, the processor is involved only at the beginning and at the end. As the I/O devices can now transmit their data or interact with the memory directly, the process is called DMA.

4. Interrupt initiated I/O.

Ans :

- In programmed initiated, CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.
- This is a time consuming process since it keeps the processor busy needlessly.
- It can be avoided by using an interrupt facility and a special command to inform the interface to issue an interrupt request signal when data are available from the device.
- In the meantime CPU can proceed to execute another program.
- The interface meanwhile keeps monitoring the device.
- When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.
- While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- The CPU deviates from what it is doing to take care of the input or output transfer.
- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

5. What is priority interrupt?

Ans :

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt.

6. What is computer bus?

Ans :

A bus in computer terminology represents a physical connection used to carry a signal from one point to another. The signal carried by a bus may represent address, data, control signal, or power. Typically, a bus consists of a number of connections running together. Each connection is called a bus line. A bus line is normally identified by a number. Related groups of bus lines are usually identified by a name. For example, the group of bus lines 1 to 16 in a given computer system may be used to carry the address of memory locations, and therefore are identified as address lines.

Depending on the signal carried, there exist at least four types of buses: address, data, control, and power buses. Data buses carry data, control buses carry control signals, and power buses carry the power-supply/ground voltage. A bus can be synchronous if data transfer over the bus is controlled by a bus clock. The clock acts as the timing reference for all bus signals. A bus is asynchronous if data transfer over the bus is based on the availability of the data and not on a clock signal. Data is transferred over an asynchronous bus using a technique called handshaking.

7. Bus Protocols

Ans :

A bus is a communication channel shared by many devices and hence rules need to be established in order for the communication to happen correctly. These rules are called bus protocols. Design of a bus architecture involves several tradeoffs related to the width of the data bus, data transfer size, bus protocols, clocking, etc. Depending on whether the bus transactions are controlled by a clock or not, buses are classified into synchronous and asynchronous buses. Depending on whether the data bits are sent on parallel wires or multiplexed onto one single wire, there are parallel and serial buses. Control of the bus communication in the presence of multiple devices necessitates defined procedures called arbitration schemes.

8. Bus Arbitration

Ans :

Bus arbitration is needed to resolve conflicts when two or more devices want to become the bus master at the same time. In short, arbitration is the process of selecting the next bus master from among multiple candidates. Conflicts can be resolved based on fairness or priority in a centralized or distributed mechanisms. Centralized Arbitration In centralized arbitration schemes, a single arbiter is used to select the next master. A simple form of centralized arbitration uses a bus request line, a bus grant line, and a bus busy line. Each of these lines is shared by potential masters, which are daisy-chained in a cascade. Figure (a) shows this simple centralized arbitration scheme. In the figure, each of the potential masters can submit a bus request at any time.

A fixed priority is set among the masters from left to right. When a bus request is received at the central bus arbiter, it issues a bus grant by asserting the bus grant line. When the potential master that is closest to the arbiter (potential master 1) sees the bus grant signal, it checks to see if it had made a bus request. If yes, it takes over the bus and stops propagation of the bus grant signal any further. If it has not made a request, it will simple turn the bus grant signal to the next master to the right (potential master 2), and so on. When the transaction is complete, the busy line is deasserted.

9. PCI interface.

Ans :

The PCI was developed as a low-cost bus that is truly processor independent. Its design anticipated a rapidly growing demand for bus bandwidth to support high-speed disks and graphic and video devices, as well as the specialized needs of multiprocessor systems. As a result, the PCI is still popular as an industry standard almost a decade after it was first introduced in 1992. An important feature that the PCI pioneered is a plug-and-play capability for connecting I/O devices. To connect a new device, the user simply connects the device interface board to the bus. The software takes care of the rest.

10. Explain SCSI.

Ans :

The acronym SCSI stands for Small Computer System Interface. It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X3.131. In the original specifications of the standard, devices such as disks are connected to a computer via a 50- wire cable, which can be up to 25 meters in length and can transfer data at rates up to 5 megabytes/s. The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years. SCSI-2 and SCSI-3 have been defined, and each has several options.

A SCSI bus may have eight data lines, in which case it is called a narrow bus and transfers data one byte at a time. Because of these various options, the SCSI connector may have 50, 68, or 80 pins. The maximum transfer rate in commercial devices that are currently available varies from 5 megabytes/s to 160 megabytes/so The most recent version of the standard is intended to support transfer rates up to 320 megabytes/s, and 640 megabytes/s is anticipated a little later.

Devices connected to the SCSI bus are not part of the address space of the processor in the same way as devices connected to the processor bus. The SCSI bus is connected to the processor bus through a SCSI controller. This controller uses DMA to transfer data packets from the main memory to the device, or vice versa. A packet may contain a block of data, commands from the processor to the device, or status information about the device.

A controller connected to a SCSI bus is one of two types - an initiator or a target. An initiator has the ability to select a particular target and to send commands specifying the operations to be performed. The disk controller operates as a target. It carries out the commands it receives from the initiator. The initiator establishes a logical connection with the intended target. Once this connection has been established, it can be suspended and restored as needed to transfer commands and bursts of data. While a particular connection is suspended, other devices can use the bus to transfer information. This ability to overlap data transfer requests is one of the key features of the SCSI bus that leads to its high performance.

Choose the Correct Answers

1. Which of the following is an essential data transfer technique? [b]
   (a) MMA (b) DMA
   (c) CAD (d) CAM

2. The method of accessing the I/O devices by repeatedly checking the status flags is _____ [a]
   (a) Program-controlled I/O (b) Memory-mapped I/O
   (c) I/O mapped (d) None of the mentioned

3. The time between the receiver of an interrupt and its service is _____ [b]
   (a) Interrupt delay (b) Interrupt latency
   (c) Cycle time (d) Switching time

4. The DMA controller has _____ registers. [c]
   (a) 4 (b) 2
   (c) 3 (d) 1

5. The controller is connected to the _____ [b]
   (a) Processor BUS (b) System BUS
   (c) External BUS (d) None of the mentioned

6. The technique where the controller is given complete access to main memory is _____ [d]
   (a) Cycle stealing (b) Memory stealing
   (c) Memory Con (d) Burst mode

7. _____ is an extension of the processor BUS. [d
   (a) SCSI BUS (b) USB
   (c) PCI BUS (d) None of the mentioned

8. The video devices are connected to _____ BUS. [
   (a) PCI (b) USB
   (c) HDMI (d) SCSI

9. The PCI BUS has _____ interrupt request lines.
   (a) 6 (b) 1
   (c) 4 (d) 3

10. The SCSI BUS uses _____ arbitration.
   (a) Distributed (b) Centralised
   (c) Daisy chain (d) Hybrid

Fill in the Blanks

1. In the _____ data transfer scheme, data is directly transferred from an I/O device to RAM or from RAM to an I/O device.
2. The process wherein the processor constantly checks the status flags is called as _____
3. The resistor which is attached to the service line is called _____
4. An interrupt that can be temporarily ignored is _____
5. The starting address sent by the device in vectored interrupt is called as _____
6. The DMA transfers are performed by a control circuit called as _____
7. The technique whereby the DMA controller steals the access cycles of the processor to operate is called _____
8. SCSI stands for _____
9. The best mode of connection between devices which need to send or receive large amounts of data over a short distance is _____
10. The key feature of the PCI BUS is _____

Answers

1. DMA
2. Polling
3. Pull-up resistor
4. Maskable interrupt
5. Interrupt vector
6. DMA controller
7. Cycle stealing
8. Small Computer System Interface
9. Parallel port
10. Plug and Play capability