

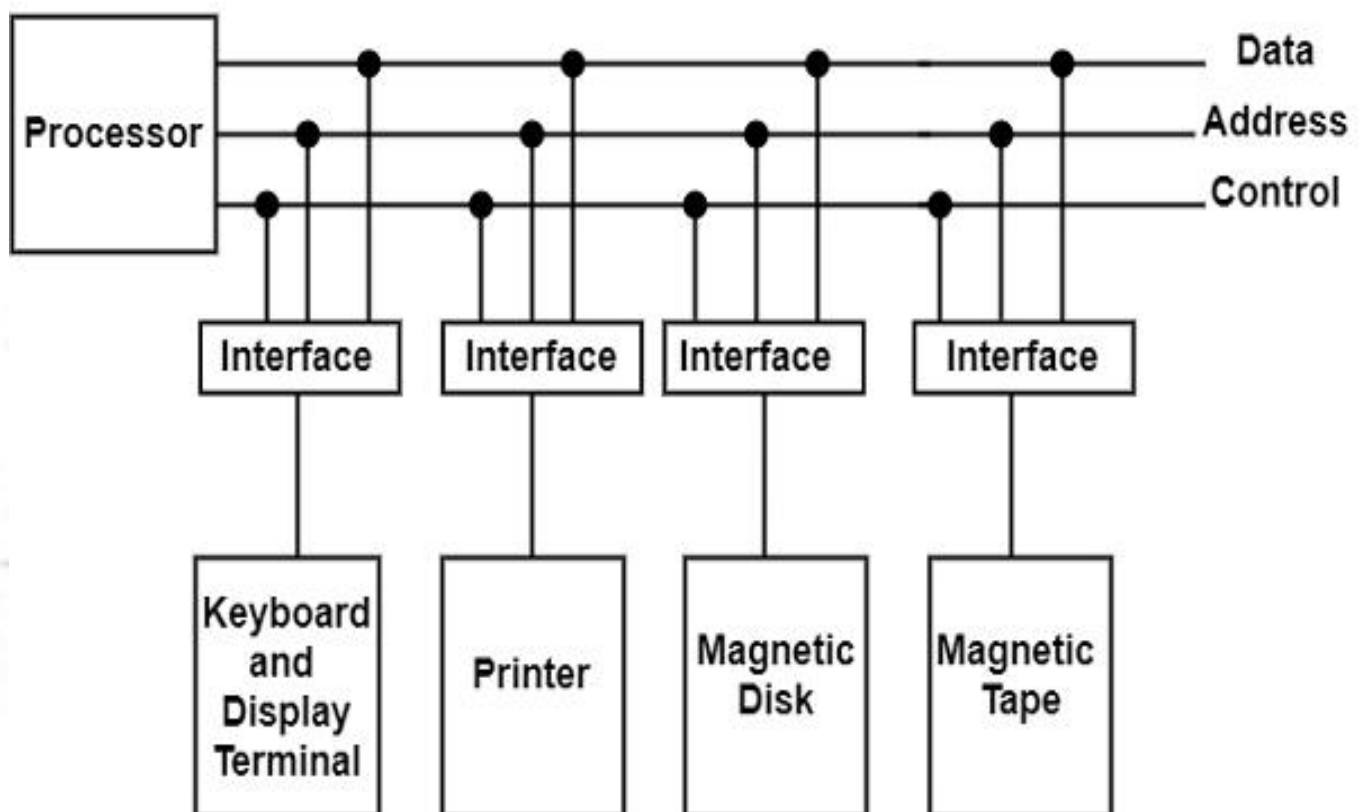
Input-Output Interface:-

The I/O interface supports a method by which data is transferred between internal storage and external I/O devices. All the peripherals connected to a computer require special communication connections for interfacing them with the CPU.

I/O Bus and Interface Modules

The I/O bus is the route used for peripheral devices to interact with the computer processor. A typical connection of the I/O bus to I/O devices is shown in the figure.

Connection of I/O Bus to I/O Devices



The I/O bus includes data lines, address lines, and control lines. In any general-purpose computer, the magnetic disk, printer, and keyboard, and display terminal are commonly employed. Each peripheral unit has an interface unit associated

with it. Each interface decodes the control and address received from the I/O bus. It can describe the address and control received from the peripheral and supports signals for the peripheral controller. It also conducts the transfer of information between peripheral and processor and also integrates the data flow.

The I/O bus is linked to all peripheral interfaces from the processor. The processor locates a device address on the address line to interact with a specific device. Each interface contains an address decoder attached to the I/O bus that monitors the address lines.

When the address is recognized by the interface, it activates the direction between the bus lines and the device that it controls. The interface disables the peripherals whose address does not equivalent to the address in the bus.

An interface receives any of the following four commands –

Control – A control command is given to activate the peripheral and to inform its next task. This control command depends on the peripheral, and each peripheral receives its sequence of control commands, depending on its mode of operation.

Status – A status command can test multiple test conditions in the interface and the peripheral.

Data Output – A data output command creates the interface counter to the command by sending data from the bus to one of its registers.

Data Input – The data input command is opposite to the data output command. In data input, the interface gets an element of data from the peripheral and places it in its buffer register.

Asynchronous Data Transfer:-

We know that, the internal operations in individual unit of digital system are synchronized by means of clock pulse, means clock pulse is given to all registers within a unit, and all data transfer among internal registers occur simultaneously during occurrence of clock pulse. Now, suppose any two units of digital system are designed independently such as CPU and I/O interface.

And if the registers in the interface (I/O interface) share a common clock with CPU registers then transfer between the two units is said to be synchronous. But in most cases, the internal timing in each unit is independent from each other in such a way that each uses its own private clock for its internal registers. In that case, the two units are said to be asynchronous to each other, and if data transfer occur between them this data transfer is said to be Asynchronous Data Transfer.

But, the Asynchronous Data Transfer between two independent units requires that control signals be transmitted between the communicating units so that the time can be indicated at which they send data.

This asynchronous way of data transfer can be achieved by two methods:

- One way is by means of strobe pulse which is supplied by one of the units to other unit. When transfer has to occur. This method is known as “Strobe Control”.
- Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another signal to acknowledge receipt of the data. This method of data transfer between two independent units is said to be “Handshaking”.

The strobe pulse and handshaking method of asynchronous data transfer are not restricted to I/O transfer. In fact, they are used extensively on numerous occasion requiring transfer of data between two independent units. So, here we consider the transmitting unit as source and receiving unit as destination.

As an example: The CPU is the source during an output or write transfer and is the destination unit during input or read transfer.

And thus, the sequence of control during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination.

So, while discussing each way of data transfer asynchronously we see the sequence of control in both terms when it is initiated by source or when it is initiated by destination. In this way, each way of data transfer, can be further divided into parts, source initiated and destination initiated.

We can also specify asynchronous transfer between two independent units by means of a timing diagram that shows the timing relationship that exists between the control and the data buses.

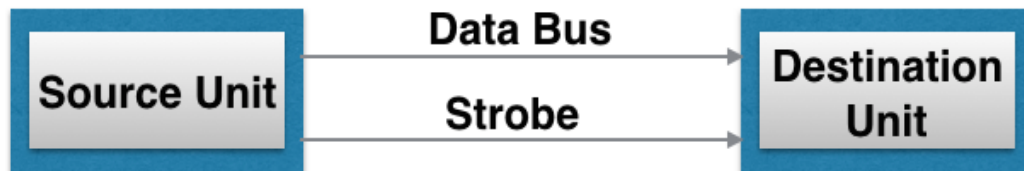
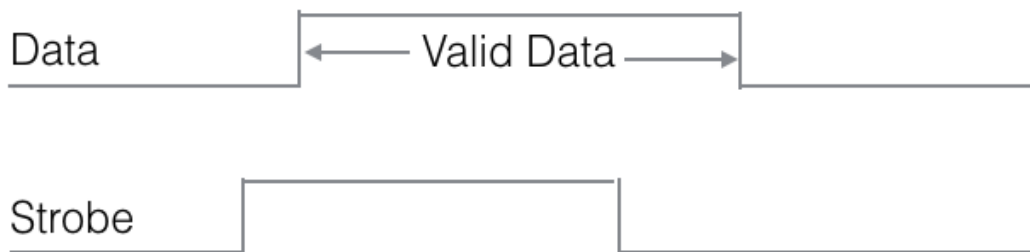
Now, we will discuss each method of asynchronous data transfer in detail one by one.

1. Strobe Control:

The Strobe Control method of asynchronous data transfer employs a single control line to time each transfer. This control line is also known as strobe and it may be achieved either by source or destination, depending on which initiate transfer.

Source initiated strobe for data transfer:

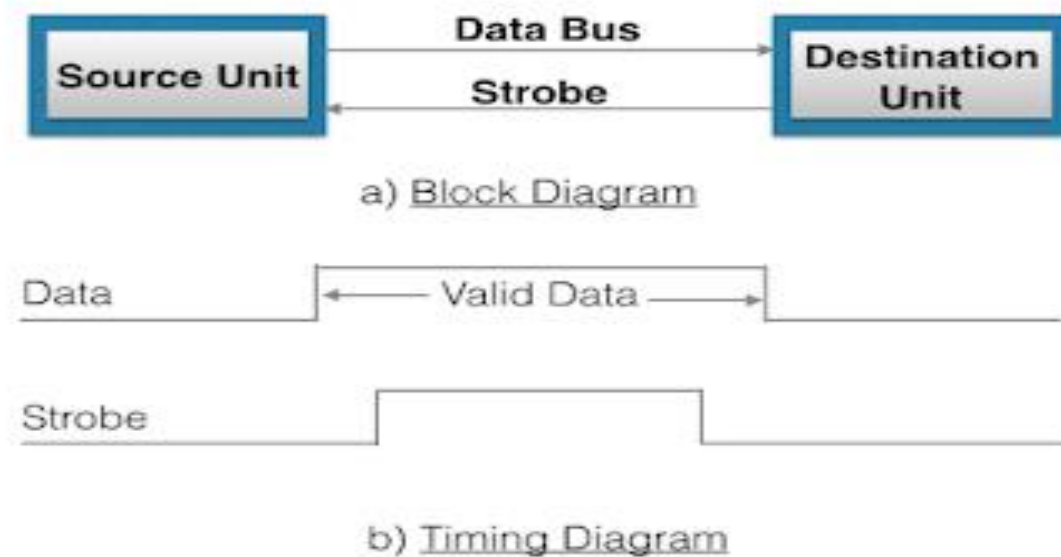
The block diagram and timing diagram of strobe initiated by source unit is shown in figure below:

a) Block Diagramb) Timing Diagram

In block diagram we see that strobe is initiated by source, and as shown in timing diagram, the source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates a strobe pulse. The information on data bus and strobe control signal remain in the active state for a sufficient period of time to allow the destination unit to receive the data. Actually, the destination unit uses a falling edge of strobe control to transfer the contents of data bus to one of its internal registers. The source removes the data from the data bus after it disables its strobe pulse. New valid data will be available only after the strobe is enabled again.

Destination-initiated strobe for data transfer:

The block diagram and timing diagram of strobe initiated by destination is shown in figure below:



In block diagram, we see that, the strobe initiated by destination, and as shown in timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it. The falling edge of strobe pulse can be used again to trigger a destination register. The destination unit then disables the strobe. And source removes the data from data bus after a per determine time interval.

Now, actually in computer, in the first case means in strobe initiated by source - the strobe may be a memory-write control signal from the CPU to a memory unit. The source, CPU, places the word on the data bus and informs the memory unit, which is the destination, that this is a write operation.

And in the second case i.e., in the strobe initiated by destination - the strobe may be a memory read control from the CPU to a memory unit. The destination, the CPU, initiates the read operation to inform the memory, which is a source unit, to place selected word into the data bus.

2. Handshaking:

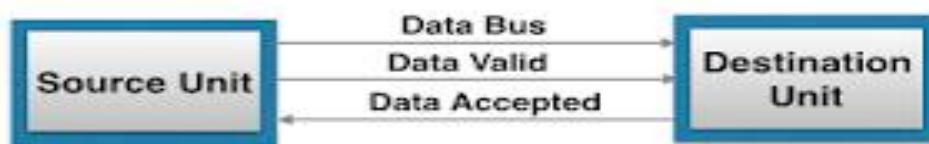
The disadvantage of strobe method is that source unit that initiates the transfer has no way of knowing whether the destination has actually received the data that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit, has actually placed data on the bus. This problem can be solved by handshaking method.

Hand shaking methods introduce a second control signal line that provides a replay to the unit that initiates the transfer.

In it, one control line is in the same direction as the data flow in the bus from the source to destination. It is used by source unit to inform the destination unit whether there are valid data in the bus. The other control line is in the other direction from destination to the source. It is used by the destination unit to inform the source whether it can accept data. And in it also, sequence of control depends on unit that initiate transfer. Means sequence of control depends whether transfer is initiated by source and destination. Sequences of control in both of them are described below:

Source initiated Handshaking:

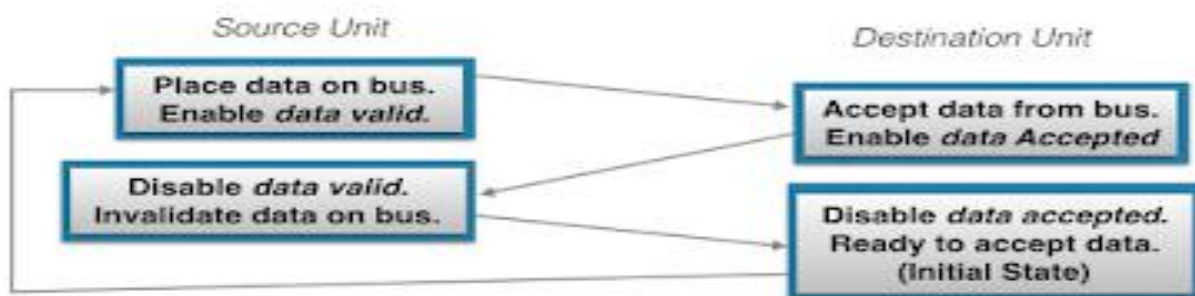
The source initiated transfer using handshaking lines is shown in figure below



a) Block Diagram



b) Timing Diagram



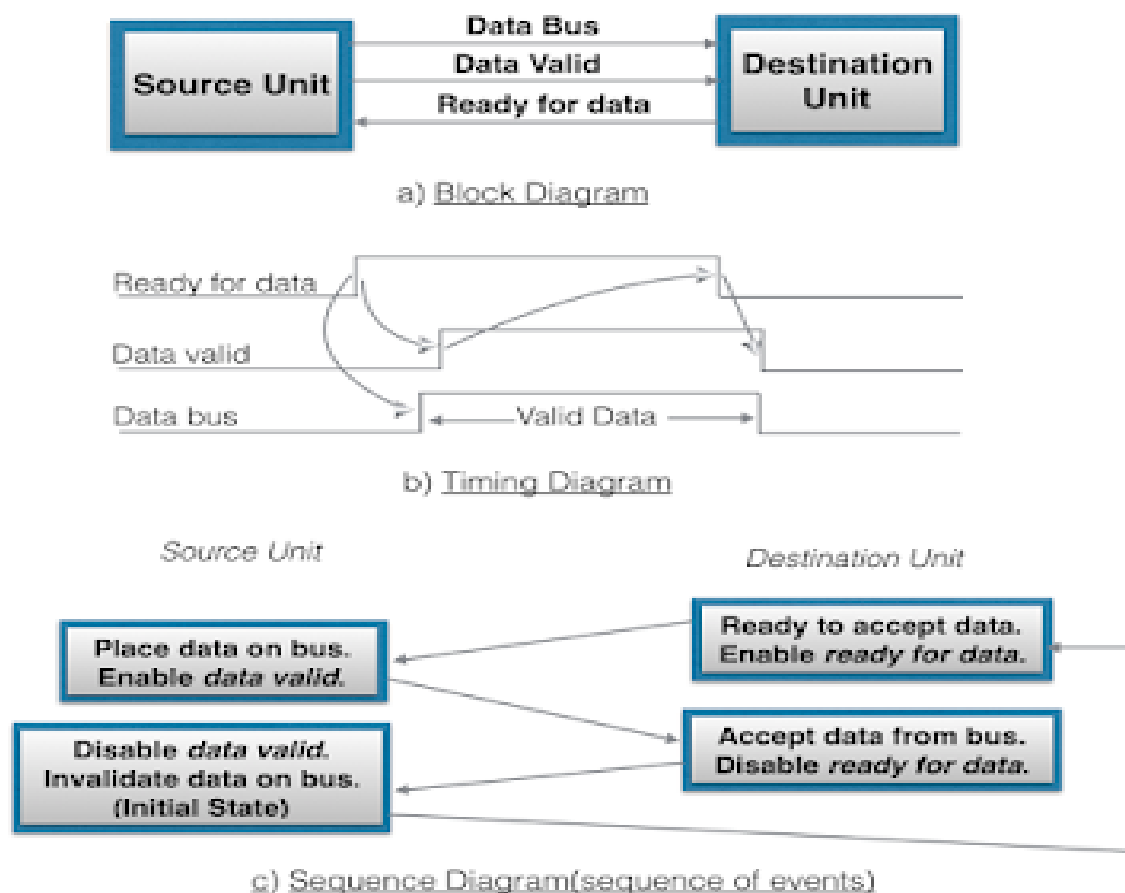
c) Sequence Diagram (Sequence of events)

In its block diagram, we see that two handshaking lines are "data valid", which is generated by the source unit, and "data accepted", generated by the destination unit.

The timing diagram shows the timing relationship of exchange of signals between the two units. Means as shown in its timing diagram, the source initiates a transfer by placing data on the bus and enabling its data valid signal. The data accepted signal is then activated by destination unit after it accepts the data from the bus. The source unit then disables its data valid signal which invalidates the data on the bus. After this, the destination unit disables its data accepted signal and the system goes into initial state. The source unit does not send the next data item until after the destination unit shows its readiness to accept new data by disabling the data accepted signal. This sequence of events described in its sequence diagram, which shows the above sequence in which the system is present, at any given time.

Destination initiated handshaking:

The destination initiated transfer using handshaking lines is shown in figure below:



In its block diagram, we see that the two handshaking lines are "data valid", generated by the source unit, and "ready for data" generated by destination unit. Note that the name of signal data accepted generated by destination unit has been changed to ready for data to reflect its new meaning.

In it, transfer is initiated by destination, so source unit does not place data on data bus until it receives ready for data signal from destination unit. After that, hand shaking process is same as that of source initiated.

The sequence of event in it is shown in its sequence diagram and timing relationship between signals is shown in its timing diagram. Thus, here we can say that, sequence of events in both cases would be identical. If we consider ready for data signal as the complement of data accept. Means, the only difference between source and destination initiated transfer is in their choice of initial state.

Mode of Transfer:-

The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes.

Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed I/O.
2. Interrupt- initiated I/O.
3. Direct memory access (DMA).

Programmed I/O:- It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and

memory. In this case it requires constant monitoring by the CPU of the peripheral devices.

Interrupt- initiated I/O:- Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

Note:

Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the processor to transfer data between memory and the I/O module, and any data transfer must transverse a path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.

1. The I/O transfer rate is limited by the speed with which the processor can test and service a device.
2. The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.

Direct Memory Access:- The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

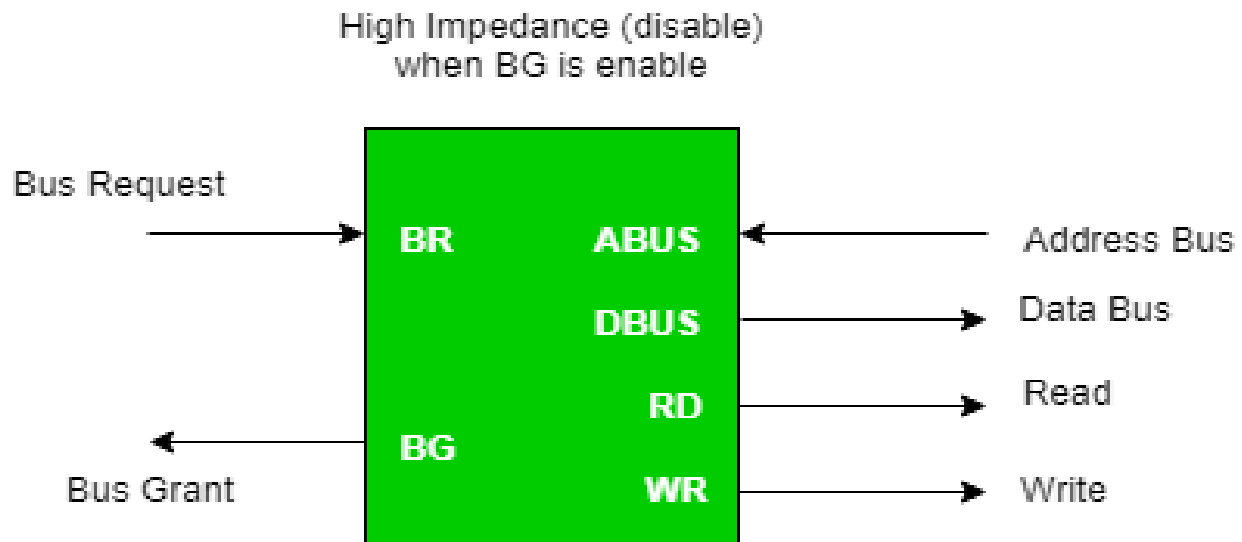


Figure - CPU Bus Signals for DMA Transfer

Bus Request: - It is used by the DMA controller to request the CPU to relinquish the control of the buses.

Bus Grant: - It is activated by the CPU to inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

Thank You