GraphicEra
(Deemed to be University)
Accredited by NAAC with Grade A

MCA I/M.Sc. (IT) I

Computer Organization and Architecture

TMC 102/TMI 102

By

Jaishankar Bhatt

Assistant Professor

Graphic Era Deemed to be University

Dehradun

Unit 3

Table of Contents

Input Output Interface: - Input-output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral. The major differences are:

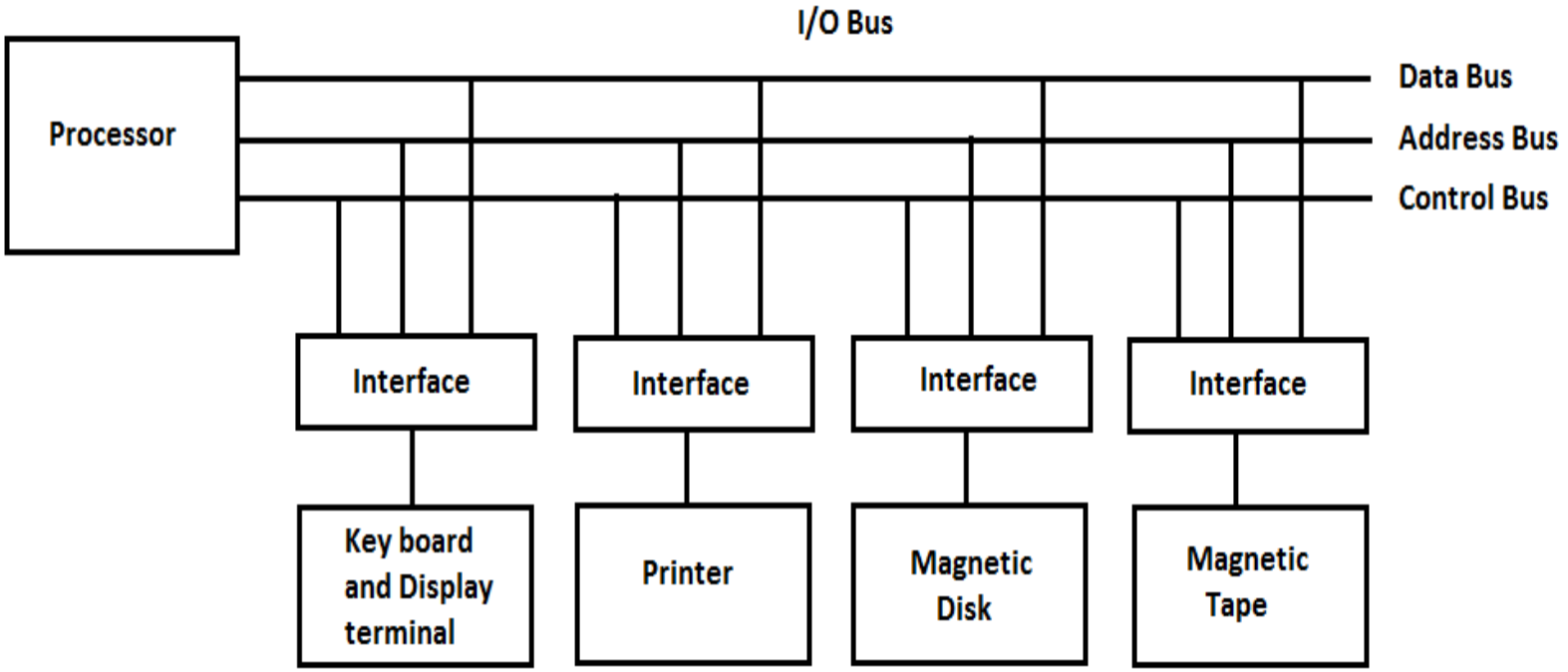
1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.

2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be need.

3. Data codes and formats in peripherals differ form the word format in the CPU and memory.

4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device. In addition, each device may have its own controller that supervises the operations of the particular mechanism in the peripheral.



Block diagram of Input Output interface

Modes of Transfer: - There are various techniques which are used to transfer data from peripheral device to memory, generally it is based on the architecture of the microprocessor. Information is stored in memory for later processing. Data transfer between the central computer and I/O devices may be handled in a variety of modes. Some modes use the CPU as an intermediate path; other transfers the data directly to and from the memory unit. In a basic computer data transfer to and from peripherals may be handled in one of three possible modes:

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

1. **Programmed I/O: -** CPU is the master of system buses. In programmed I/O CPU supervises the data transfer. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. In the programmed I/O method, the 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.
2. **Interrupt Initiated I/O: -** In the programmed I/O mode CPU wastes its time in monitoring of I/O devices. It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device. In the meantime the CU 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. Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.

3. Direct Memory Access (DMA): - In direct memory access (DMA), the interface transfers data into and out of the memory unit 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, the DMA transfers the data directly into memory. The CPU merely delays its memory access operation to allow the direct memory I/O transfer.

Interrupts: -The term interrupt loosely is used for any exceptional event that causes temporary transfer of control of CPU from one program to the other which is causing the interrupt. Interrupts are primarily issued on:

i. Initiation of Input/output operation

ii. Completion of an Input/output operation

Iii. Occurrence of hardware or software errors.

Interrupts can he generated by various sources internal external to the CPU. An interrupt generated internally by CPU is sometimes termed as Traps. The traps are normally result of programming errors such as division by zero while execution of a program. The two key issues in Interrupt driven Input/output are:

1. To determine the device which has issued an interrupt

2. In case of occurrence of multiple interrupts which one to be processed first

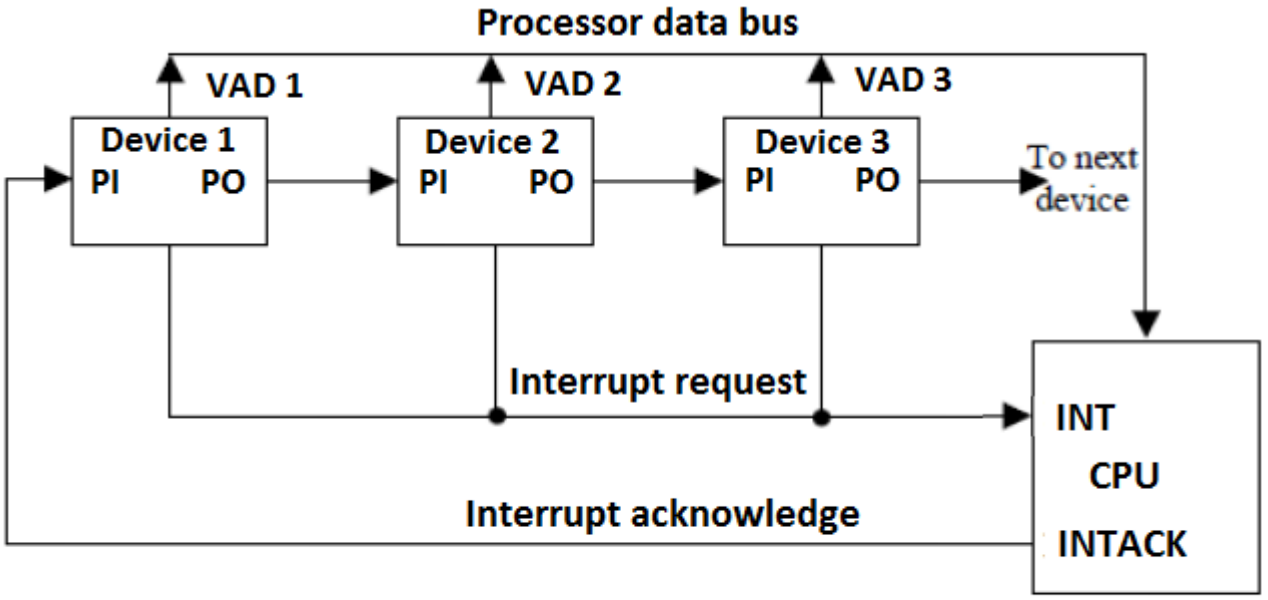
Priority Interrupt: - Data transfer between the CPU and an I/O device is initiated by the CPU. However, the CPU cannot start the transfer unless the device is ready to communicate with the CPU. The readiness of the device can be determined from an interrupt signal.

In a typical application a number of I/O devices are attached to the computer, with each device being able to originate an interrupt request. The first task of the interrupt system is to identify the source of the interrupt. There is also the possibility that several sources will request service simultaneously. In this case the system must also decide which device to service first. Generally two techniques are used to established the priority of interrupts which are

1. Polling
2. Daisy Chaining

1. **Polling: -** A polling procedure is used to identify the highest-priority source by software means. In this method there is one common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence. The order in which they are tested determines the priority of each interrupt. The highest-priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source. Otherwise, the next-lower-priority source is tested, and so on.
2. **Daisy Chaining: -** In daisy chaining, we have one interrupt acknowledge line which is chained through various interrupt devices. All the devices are arranged according to priority i.e. high speed device placed first followed by second high priority and so on. There is just one Interrupt Request line. On receiving an Interrupt Request the Interrupt Acknowledge line is activated which in turn passes this signal device by device.

The first device which has made the interrupt request thus grabs the signal and responds by putting a word which is normally an address of interrupt servicing program or a unique identifier on the data lines. This word is also referred to as interrupt vector.



Daisy Chaining Priority Interrupt

The interrupt request line is common to all devices and forms a wired logic connection. The CPU responds to an interrupt request by enabling the interrupt acknowledge line. This signal is received by device 1 at its PI (priority in) input. The acknowledge 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.

Direct Memory Access (DMA): - The DMA is a data transfer technique in which peripherals manage the memory buses for direct interaction with main memory without involving the CPU is called direct memory access (DMA). Using DMA technique large amounts of data can be transferred between memory and the peripheral without severely impacting CPU performance. During the DMA transfer, the CPU is idle and has no control of the memory buses. A DMA controller takes over the buses to manage the transfer directly between the I/O device(s) and main memory.

During DMA transfer, the CPU is idle and has no control of the memory buses. A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

During data transfer following signals are used.

Bus Request (BR): - The bus request (BR) input is used by the DMA controller to request the CPU to relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state behaves like an open circuit, which means that the output is disconnected and dies not have a logic significance.

Bus Grant (BG): - The CPU activates the Bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

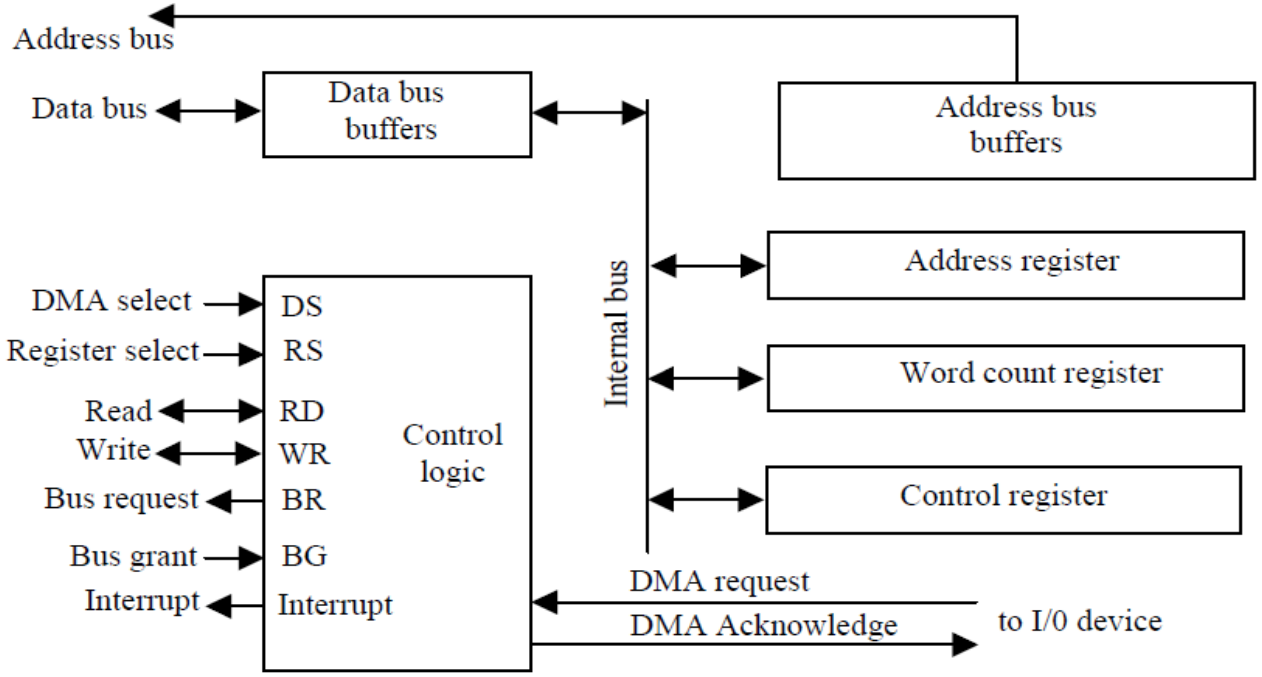
The DMA transfers the data in three modes which include the following.

a) **Burst Mode:-** In this mode DMA handover the buses to CPU only after completion of whole data transfer. Meanwhile, if the CPU requires the bus it has to stay ideal and wait for data transfer.

b) **Cycle Stealing Mode: -** In this mode, DMA gives control of buses to CPU after transfer of every byte. It continuously issues a request for bus control, makes the transfer of one byte and returns the bus. By this CPU doesn't have to wait for a long time if it needs a bus for higher priority task.

c) **Transparent Mode:-** Here, DMA transfers data only when CPU is executing the instruction which does not require the use of buses.

DMA CONTROLLER: - A DMA controller is a device, usually peripheral to a CPU that is programmed to perform a sequence of data transfers on behalf of the CPU. A DMA controller can directly access memory and is used to transfer data from one memory location to another, or from an I/O device to memory and vice versa.



DMA CONTROLLER

The DMA controller requests CPU to handle control of buses using bus request (BR) signal. The CPU grants the control of buses to DMA using bus grant (BG) signal after placing the address bus, data bus and read and write lines into high impedance state (which behave like open circuit).

CPU initializes the DMA by sending following information through the data bus.

1. Starting address of memory block for read or write operation.
2. The word count which is the number of words in the memory block.
3. Control signal to specify the mode of transfer such as read or write.
4. A control signal to start the DMA transfer.

The DMA takes control over the buses directly interacts with memory and I/O units and transfers the data without CPU intervention. When the transfer completes, DMA disables the BR line. Thus CPU disable BG line, takes control over the buses and return to its normal operation.

Working Of DMA controller: -

- The registers 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 bus grant (BG) 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 processor does not have control over the system buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.
- 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 address bits go though bus buffers into the address bus. The address register is incremented after each word that is transferred to memory.
- The word count register holds the number of words to be transferred. The register is decremented by one after each word transfer and internally tested for zero.
- The control register specifies the mode of transfer.

Advantages of DMA transfer

1. DMA speedups the Computer system performance by direct data transfers between memory and I/O devices, by bypassing the involvement of the CPU.
2. CPU is free to perform operations that do not use system buses.
3. DMA speedups the Computer system performance by bypassing the involvement of the work overload on the CPU decreases.
4. Implementing DMA also reduces the overhead of the processor

Disadvantages of DMA

1. DMA transfer requires a DMA controller to carry out the operations, hence the cost of system increases.
2. Cache coherence problem can occur while using DMA controller.

Synchronous Transfer: - The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator. Clock pulses are applied to all registers within a unit and all data transfers among internal registers occur simultaneously during the occurrence of a clock pulse. Two units, such as a CPU and an I/O interface, are designed independently of each other.

If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be **synchronous**. In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be asynchronous to each other. This approach is widely used in most computer systems.

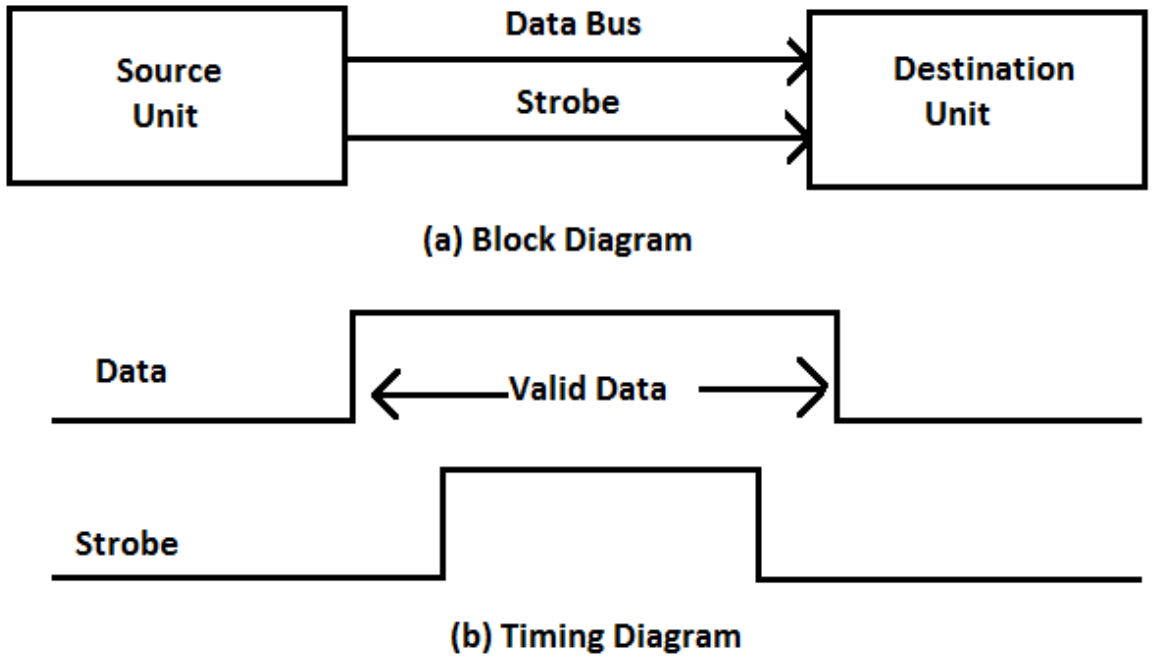
Asynchronous Data Transfer: - In computer system two units are said to be synchronous, if they are using the same clock signal. When the two units are synchronous, then both of the units are enabled and disabled at the same time. In a synchronous system every time when the sender is ready to send data, at the same time receiver unit is ready to receive the data.

If the two unit are not using the same clock signal, then the units are known as synchronous. In an asynchronous environment, whenever sender sends data to the destination unit, the destination unit or receiver is not ready to receive data.

Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. We can achieve synchronization in an asynchronous environment by using two techniques, which are as follows.

1. Strobe Control Method
2. Handshaking Method.

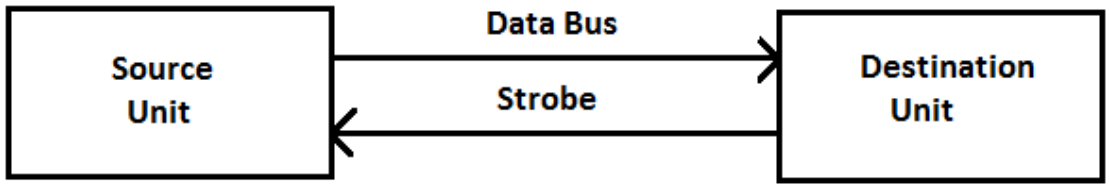
1. Strobe Control Method: - The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit.



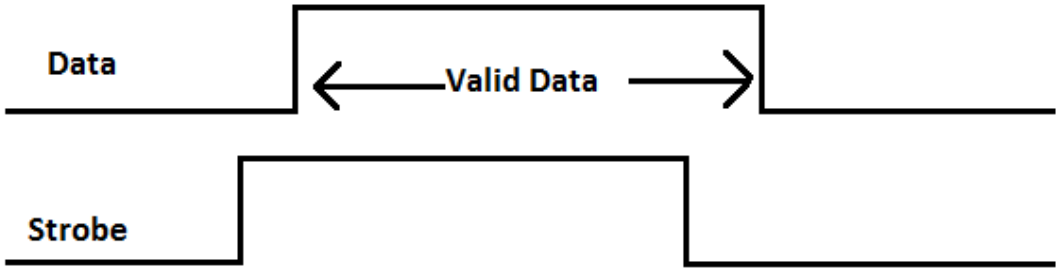
Source initiated strobe for data transfer

The source unit first places the data on the data bus. After a brief delay, the source activates the strobe pulse. The source removes the data from the bus a brief period after it disables its strobe pulse.

When the data transfer is initiated from the source unit, it is known as source initiated data transfer. The source unit first place the data on the data bus then it enables the strobe signal. The strobe signal is the indication for the destination unit that data is available on the data bus. The destination unit then remove the data from the data bus and after some time, source unit disable the strobe signal. Actually strobe is a control signal which alerts the data receiving unit that data is available on the data bus. The same procedure is repeated whenever the source needs to transfer data.



(a) Block Diagram



(b) Timing Diagram

Destination Initiated strobe for data transfer

When destination requires data, it sends a strobe signal to the source unit. After receiving the strobe signal from destination, source unit places data on the data bus and send to the destination unit. Destination unit receives data from the bus. Because the data transfer is initiated by the destination unit that's why it is called destination initiated data transfer.

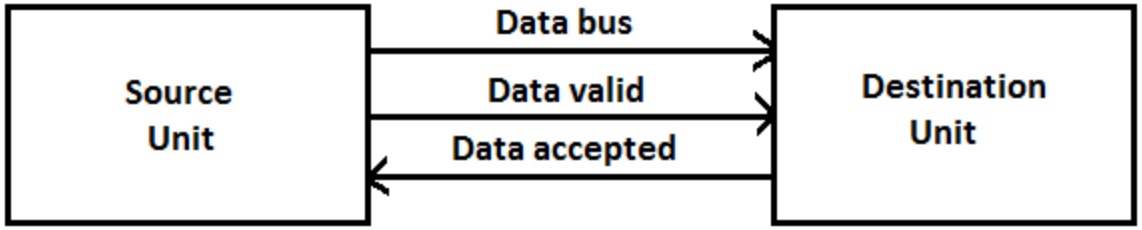
To be continued .....

Handshaking: - The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item 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 the data on the bus. The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.

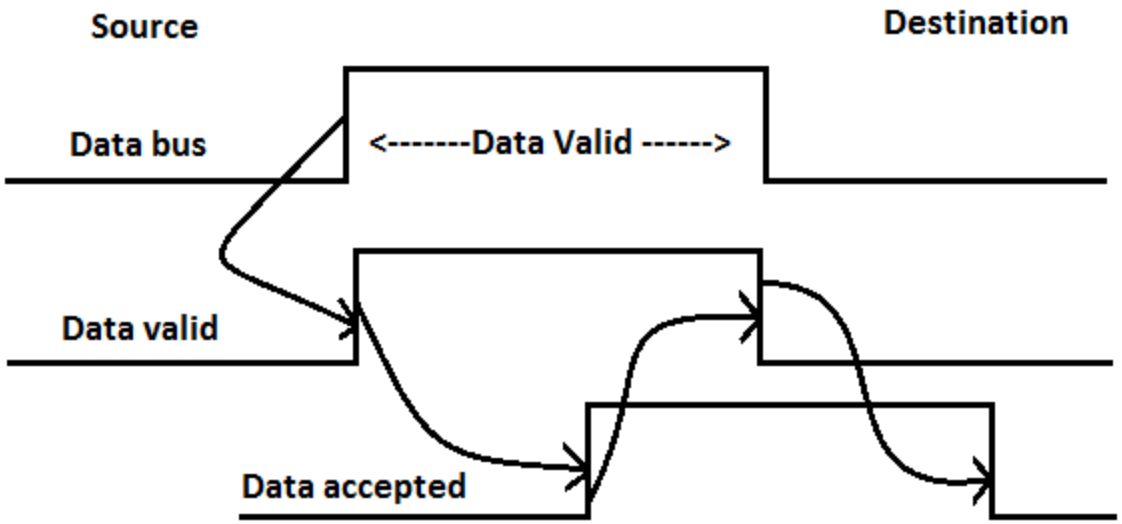
Advantages of handshaking method: -

- The handshaking method provides degree of flexibility because the successful completion of data transfer relies on active participation of both units.
- If any of the unit is faulty, the data transfer will not be completed, such an error can be detected by means of timeout mechanism which provides an alarm if the data is not completed in time.

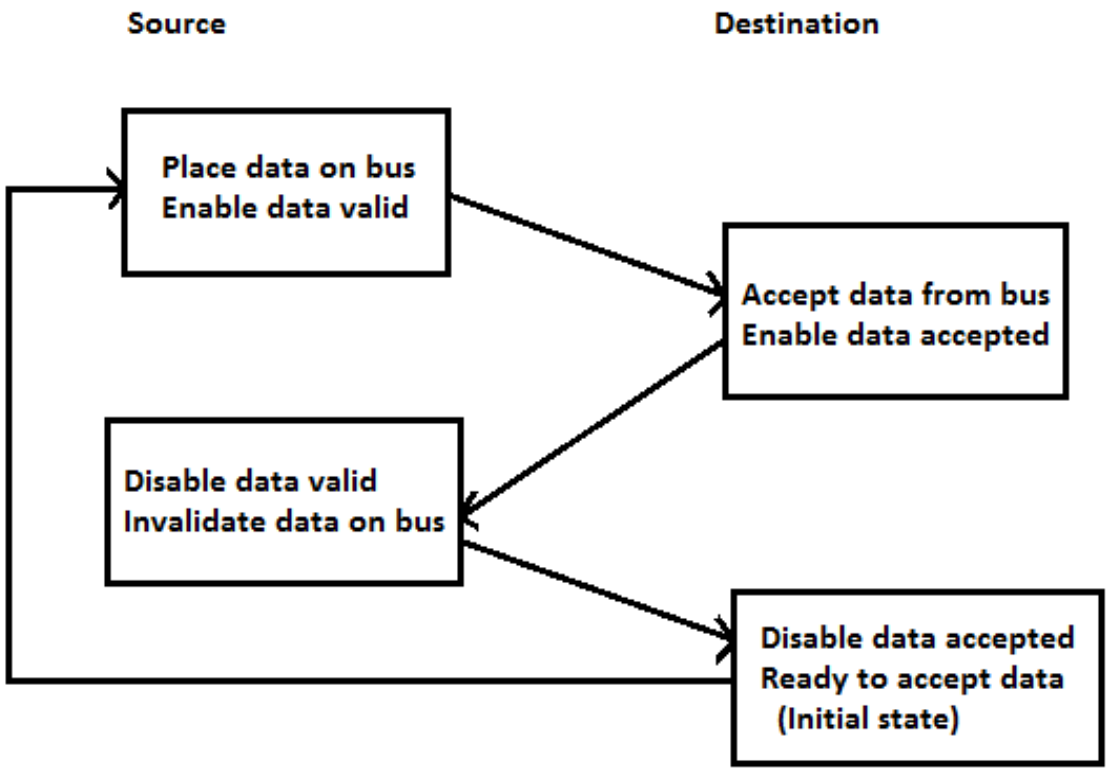
Source initiated data transfer using handshaking: - The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal. The data accepted signal is activated by the destination unit after it accepts the data from the bus. The source unit then disables its data valid signal, which invalidates the data on the bus. The destination unit then disables its data accepted signal and the system goes into its initial state. The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its data accepted signal.



(a) Block diagram



(b) Timing Diagram



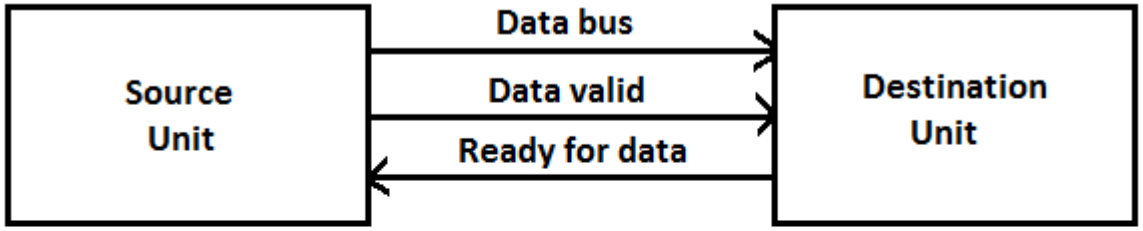
(c) Sequence of events

Source initiated data transfer using handshaking

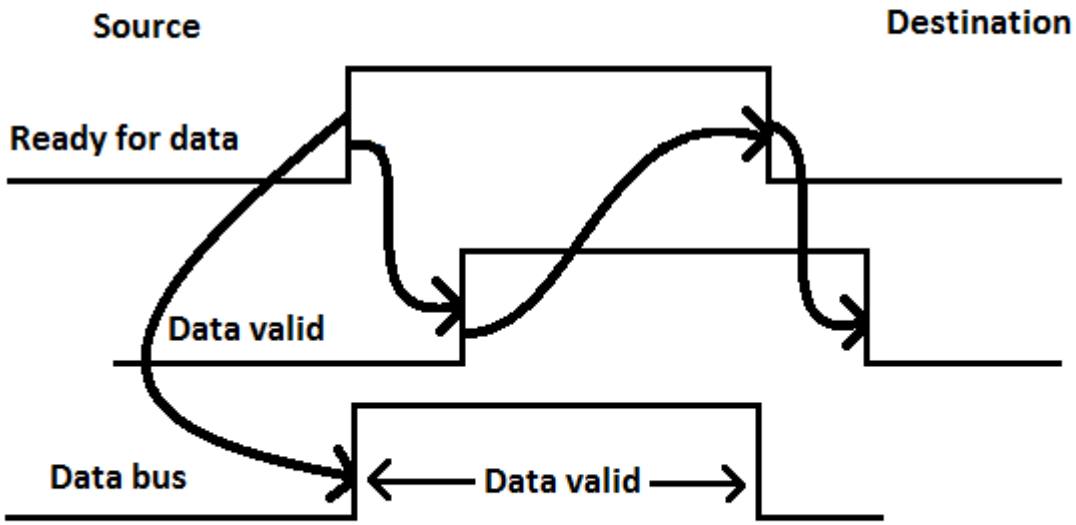
Destination initiated data transfer using handshaking: - In this method, transfer is initiated by destination, so source unit does not place data on data bus until it receives ready for data signal from destination unit. After that, hand shaking process is same as that of source initiated method.

The only difference between source initiated and destination initiated data transfer in their choice of initial state.

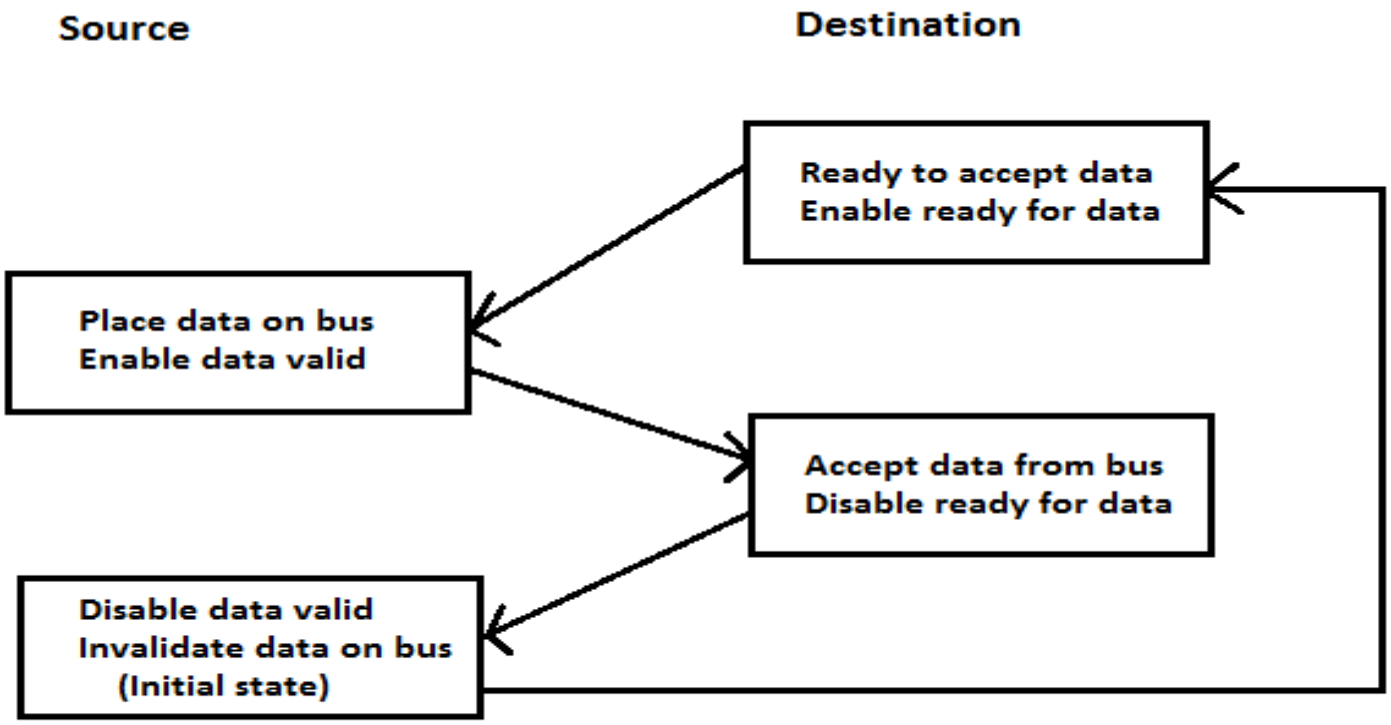
Destination initiated data transfer using handshaking: -



(a) Block diagram



(b) Timing Diagram



(c) Sequence of events

References

- Mano Morris, “Computer System Architecture”, PHI
- William Stalling, “Computer Organization & Architecture”, Pearson education Asia
- Hamacher vranesic zaky, “Computer Organization”, McGraw Hill
- B. Ram, “Computer Fundamental Architecture & Organization”, New Age.
- Tannenbaum, “Structured Computer Organization”, PHI.