30th
31st March 2020

Unit - 4

Input / Output Organization.

Topic 1: Accessing I/O Devices

A simple arrangement to connect I/O devices to a computer, is to use a single bus arrangement as shown in figure.

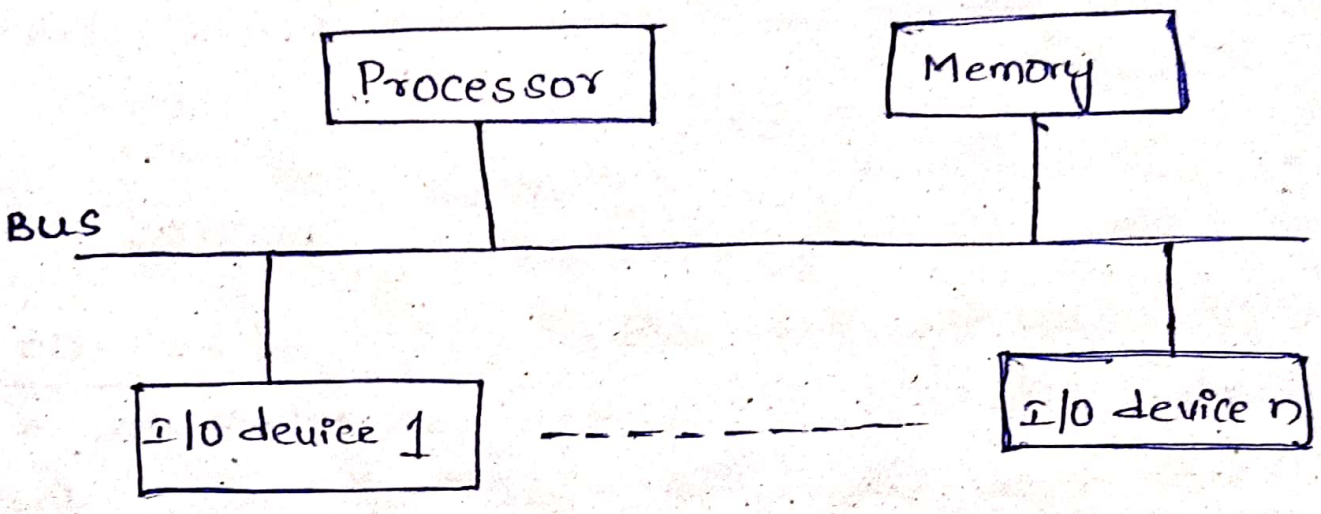


fig: A single-bus structure

→ In the above figure: The Bus enables all the devices connected to it to exchange information.
→ Typically, it consists of three sets of lines used to carry address, data and control signals.
→ Each I/O device is assigned a unique set of addresses.
→ When the processor places a particular address on the address line, the device that recognizes this address responds to the commands issued on the control lines.

→ The processor requests either a read or a write operation, and the requested data are transferred over the data lines.

→ when I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O.

→ Adv: With memory-mapped I/O, any machine instruction that accesses memory can be used to transfer data to or from an I/O device.

For ex: If DATAIN is the address of the input buffer associated with the keyboard, the instruction

MOVE DATAIN, R0

reads the data from DATAIN and stores them into processor register R0. similarly the instruction

MOVE R0, DATAOUT

sends the contents of register R0 to location DATAOUT, which may be the output data buffer of a display unit or a printer.

→ Most computer systems use memory-mapped I/O. some processors have special in and out instructions to perform I/O transfers.

For ex: processors in the intel have special I/O instructions and a separate 16-bit address space for I/O devices.

When building a computer system based on these processors, the designer has the option of connecting I/O devices to use the special I/O address space or simply incorporating them as part of the memory address space.

Input/Output Interface for an Input device :-

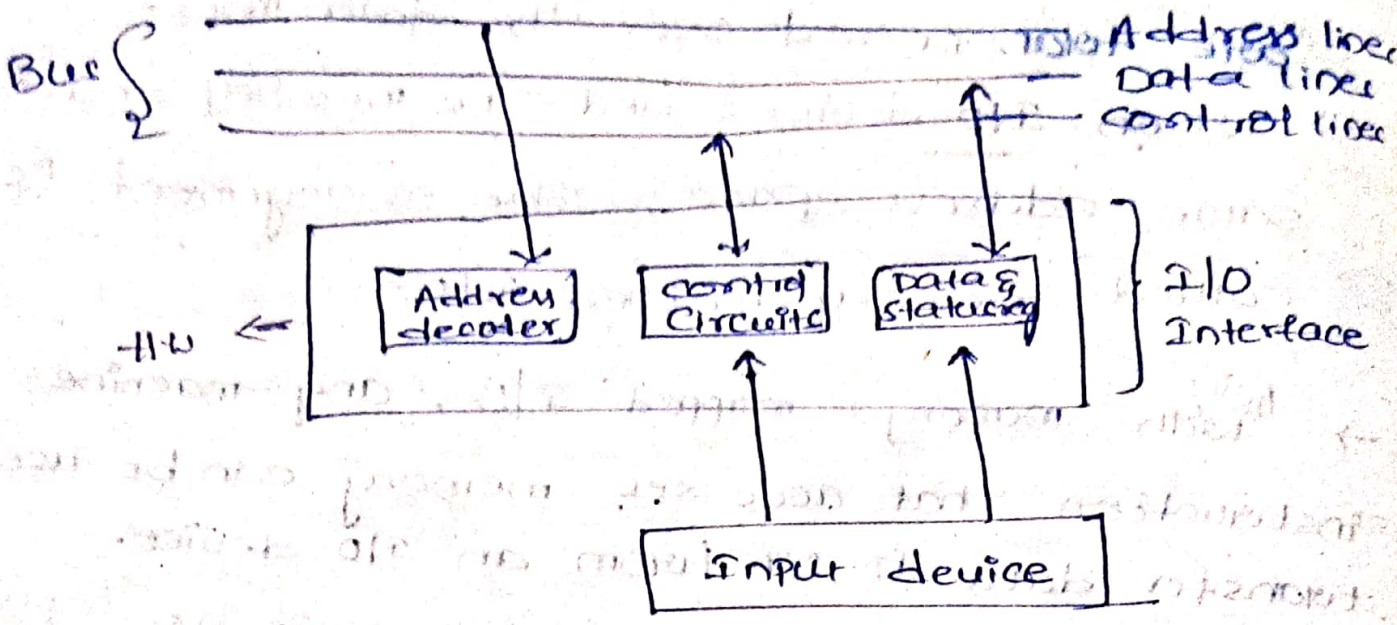


Fig: I/O interface for an input device

Fig illustrates the hardware required to connect an I/O device to the bus.

→ The address decoder enables the device to ~~connect~~ an I/O device recognize its address when this address appears on the address lines.

→ The data register holds the data being transfered to or from the processor.

→ The status register contains ilf relevant to the operation of the I/O device.

→ Both the data and status registers are connected to the data bus and assigned unique addresses.

→ The address decoder, the data and status registers and the control circuitary required to co-ordinate I/O transfers constitute the devices interface circuit.

Topic 2: Interrupts (stops the continuous progress)

Def: when a process is executed by the CPU & when a user request for another process an activity) then this will create disturbance.

Def: An interrupt is a signal sent to the processor that interrupts the current process. It may be generated by a H.W device or a software program.

Why: To improve the processing efficiency of C.P.U.

→ when a program enters a wait loop, it will repeatedly check the device status. During this period, the processor will not perform any function.

→ The interrupt request line will send a hardware signal called the interrupt signal to the processor.

→ On receiving this signal, the processor will perform the useful function during the waiting period.

→ The routine (pgm) executed in response to an interrupt request is called the interrupt-service routine, which is the PRINT routine in our example. Interrupts bear considerable resemblance to subroutine calls.

Assume that an interrupt request arrives during execution of instruction in fig:

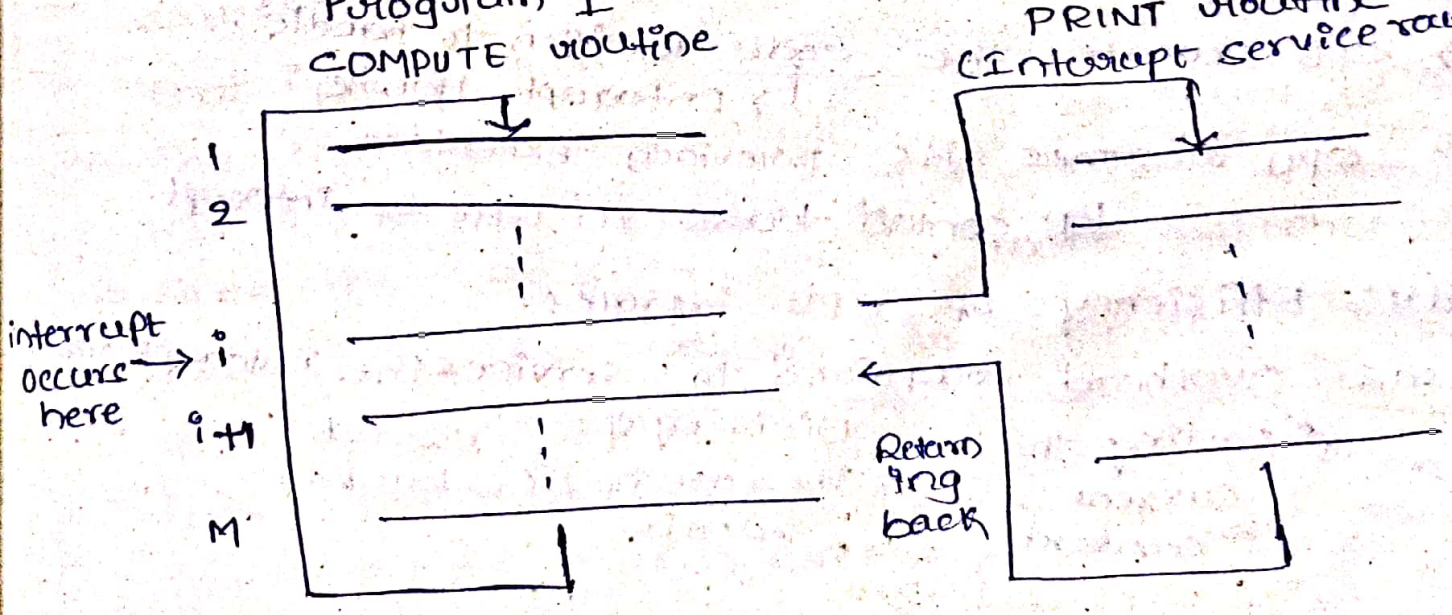


fig: Transfer of control through the use of interrupts.

→ The processor first completes execution of instruction i. Then the Program counter with the address of the first instruction of the interrupt routine.
→ After execution of the interrupt-service routine, the processor has to come back to instruction i+1.
→ Therefore, when an interrupt occurs, the current contents of the PC, which point to instruction i+1, must be put in temporary storage in a known location. A Return-from interrupt instruction at the end of the interrupt-service routine reloads the PC from the temporary storage location, causing execution to resume at instruction i+1.
→ In many processors, the returnaddress is saved on the processor stack.

1.1 Interrupt Hardware Easy Explanation for above fig.
→ Interrupt Request & Interrupt Handler (Main Points)
→ CPU executing some task.
→ User uses the keyboard to issue a high-priority cmd.
→ This issues an Interrupt Request to the CPU
→ CPU suspends the current execution of the task
→ CPU executes the code written to handle.
↳ Interrupt Handler
→ CPU resumes its previous execution.
fig. control transfer in case of Interrupt.

Adv: Efficiency of CPU improves.
Disad: Overhead required to service the Interrupt request
meaning saving the current context → Diciding the code to handle → switch back.

1.) Interrupt Hardware :-

We pointed out that an I/O device requests an interrupt by activating a bus line called Interrupt-request.

Most computers are likely to have several devices that can request an interrupt.

A single interrupt-request line may be to serve 'n' devices as

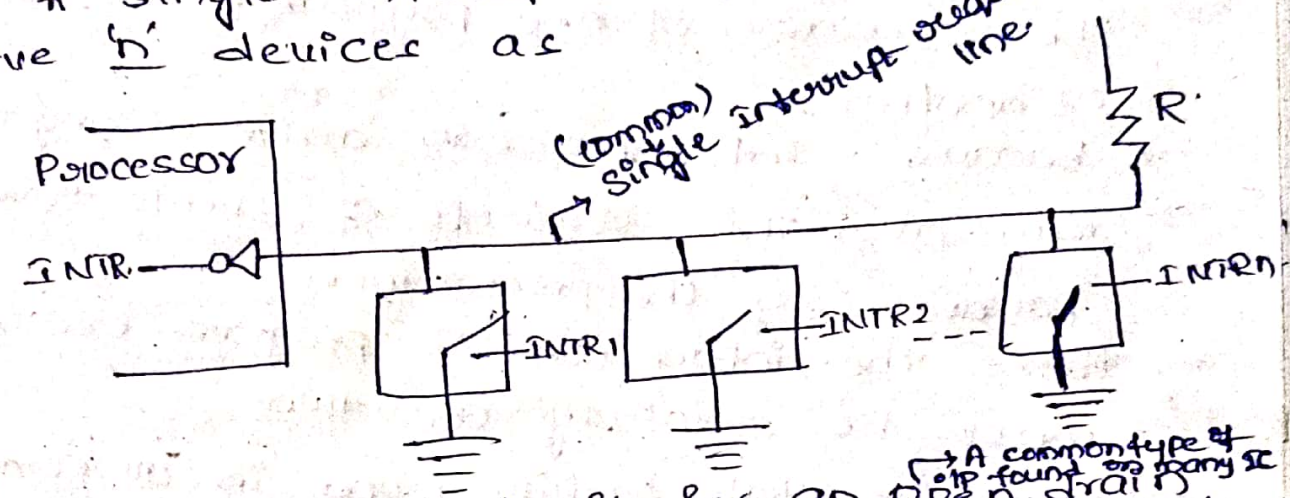


fig: An equivalent circuit for an open-drain bus used to implement a common-interrupt request line.

→ A common type of i/p found on many IC

→ All devices are connected to the line via switches to ground. To request an interrupt, a device closes its associated switch. Thus, if all interrupt-request signals INTR1 to INTRn are inactive, that is, if all switches are open, the voltage on the interrupt-request line will be equal to Vdd. This is the inactive state of the line. → supply voltages in a given circuit

→ Since the closing of one or more switches will cause the line voltage to drop to 0, the value of INTR is the logical OR of the requests from individual devices, that is,

$$INTR = INTR1 + INR2 + --+ INTRn$$

→ It is customary to use the complemented form INTR, to name the interrupt-request signal on the common line, bcz this signal is active when in the low-voltage state.

1.2 Enabling And Disabling Interrupts

→ The facilities provided in a computer must give the programmer complete control over the events that take place during program execution.

→ The arrival of an interrupt request from an external device causes the processor to suspend the execution of one program start the execution of another.

→ Because interrupts can arrive at any time, they may alter the sequence of events from that predict by the programmer.

→ Hence the interruption of program execution must be carefully controlled.

→ A fundamental facility found in all computers is the ability to enable or disable such interrupts as desired.

→ There are many situations in which the processor should ignore the interrupt requests.

For ex: in the case of computer-Print Pgm of above fig an interrupt request from the printer should be accepted only if there are o/p lines to be printed. After printing the last line of a set of n lines, interrupts should be disabled until another set becomes available for printing.

→ When an interrupt is being processed. The interrupt occuring during this time remains Pending and will be checked by processor after it has enabled interrupts.

1.3 Handling Multiple Devices (or) (Multiple Interrupts)

Let us now consider the situation where a no. of devices capable of initiating interrupts are connected to the processor.

Way 1: Disable Multiple Interrupt

↓

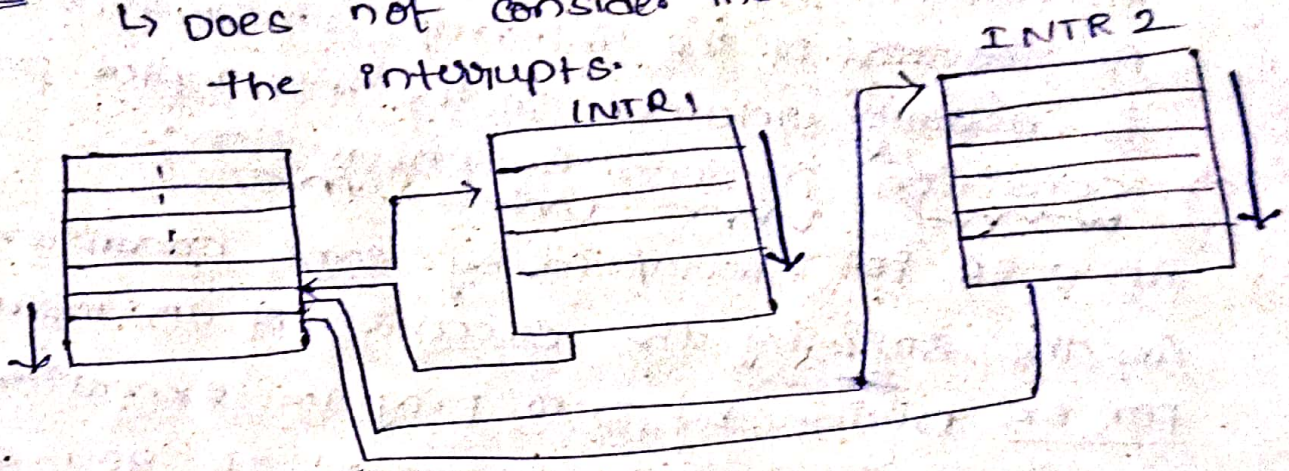
① Ignore the Interrupt Request for the time when interrupts are disabled.

↳ The interrupt occurring during this time remains pending and will be checked by processor after it has enabled interrupts.

② Sequential Execution of Interrupt

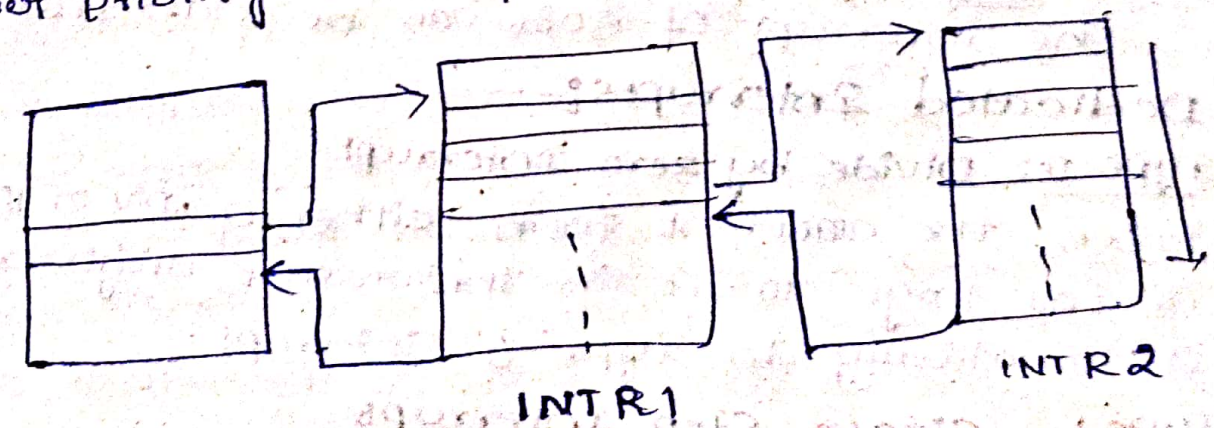
Disad: sequential nature of servicing various interrupts.

↳ Does not consider the priorities of the interrupts.



Way 2: Give Priorities to interrupts.

↳ Allow higher priority interrupts to cause lower priority interrupts to be interrupts.



Topic 3: Processor Examples

→ Let us see the interrupt structures for commonly used processors.

Interrupt structure of 8086

An 8086 interrupt can come from any one of the three sources:

a) External signal (Hardware Interrupt): An 8086 can get interrupt from an external signal applied to the nonmaskable Interrupt (NMI) input pin, OR (→ The programmer can not control → power failure / Parity Error) the Interrupt (INTR) input pin.

b) Special Instruction: 8086 supports a special instruction, INT to execute special program. At the end of the interrupt service routine, execution is usually returned to the interrupted program.

c) condition produced by Instruction:-

An 8086 is interrupted by some condition produced in the 8086 by the execution of an instruction. For ex: Divide by zero: program execution will automatically be interrupted if you attempt to divide an operand by zero.

The Fig: shows the interrupt vector table for 8086. The interrupt of 8086 can be classified as:

Dedicated Interrupts:-

Type 0: Divide by zero Interrupt

When the quotient from either a DIV or IDIV instruction is too large to fit in the result register; 8086 will automatically do type 0 interrupt.

Type 1: Single Step Interrupt

The type1 interrupt is the single step trap. In the single step mode, system will execute one instruction and wait for further direction from user.

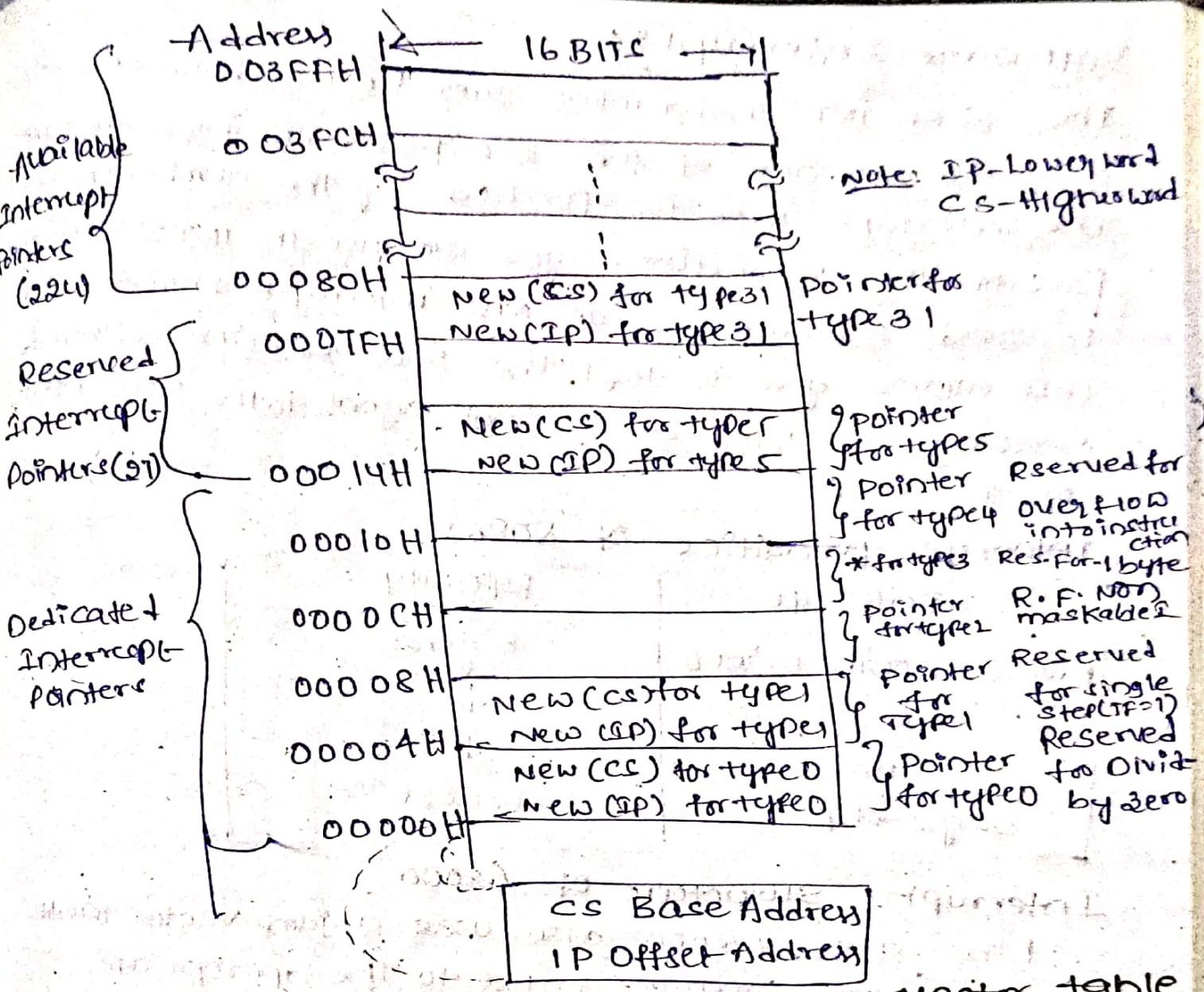


fig: 8086 Interrupt vector table

Type 2: Dedicated Non-maskable Interrupt

This interrupt can not be disabled by any S/W instruction. This interrupt is activated by low to high transition on 8086 NMI input pin. In response 8086 will do a type 2 interrupt.

Type 3: Break point:

The type3 interrupt is used to implement BREAKPOINT function in the system.

Type 4: Overflow Interrupt.

is used to check overflow condition after any signed arithmetic operation in the system. The 8086 overflow flag OF will be represented in the memory location.

Software Interrupts:-

The 8086 INT instruction can be used to cause the 8086 to do one of the 256 possible interrupt types. The interrupt type is specified by the number as a part of the instruction. you can use an INT 2 instruction to send execution to an NMI interrupt service routine. This allows you to test the NMI routine without needing to apply an external signal to the NMI input of the 8086.

Interrupt priorities of 8086:

Interrupt	Priority
Divide Error, int n, int 0	Highest
NMI	↓
INT	↓
single-step	Lowest

Interrupt structure of 68000

Like 8086, 68000 also uses a jump vector table to transfer program control to the appropriate interrupt service pgm. Whenever an exception occurs. The 68000 can be operated in either supervisory mode or user mode. A no. of instructions called Privileged instructions can be used only in supervisory mode. An attempt to execute these instructions in user mode results in a privilege violation which is one type of exception.

Like this there are no. of events that can generate an exception in 68000. To process exception the 68000 must be in supervisor mode.

TOPIC 4: Direct Memory Access.

A special control unit may be provided to allow the transfer of large block of data at high speed directly between the External device and main memory, with out continous intervention by the processor. This approach is called "DMA".

→ The transfer of a block of words, the processor sends
Starting address
Number of words in the block
Direction of transfer.

→ When a block of data is transferred, the DMA controller increment the memory address for successive words and keep track of number of words and it also informs the processor by raising an interrupt signal.

→ While DMA control is taking place, the Pgm requested the transfer can not continue and the Processor can be used to execute another program

→ After DMA transfer is completed, the processor returns to the program the requested the transfer.

R/W Determines the direction of transfer when R/W = 1, DMA controller read data from memory to I/O device.

R/W = 0, DMA controller perform write operation.

Done Flag = 1, the controller has completed transferring a block of data & is ready to receive another command.

IE = 1, it causes the controller to raise an interrupt after it has completed transferring the block of data.
Interrupt Enable

IRQ = 1, it indicates that the controller has requested an INR

Registers used by DMA controller:-

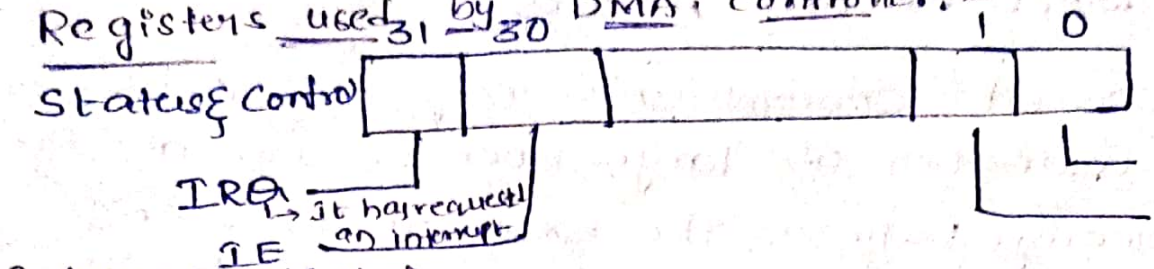


fig: Registers in DMA interface

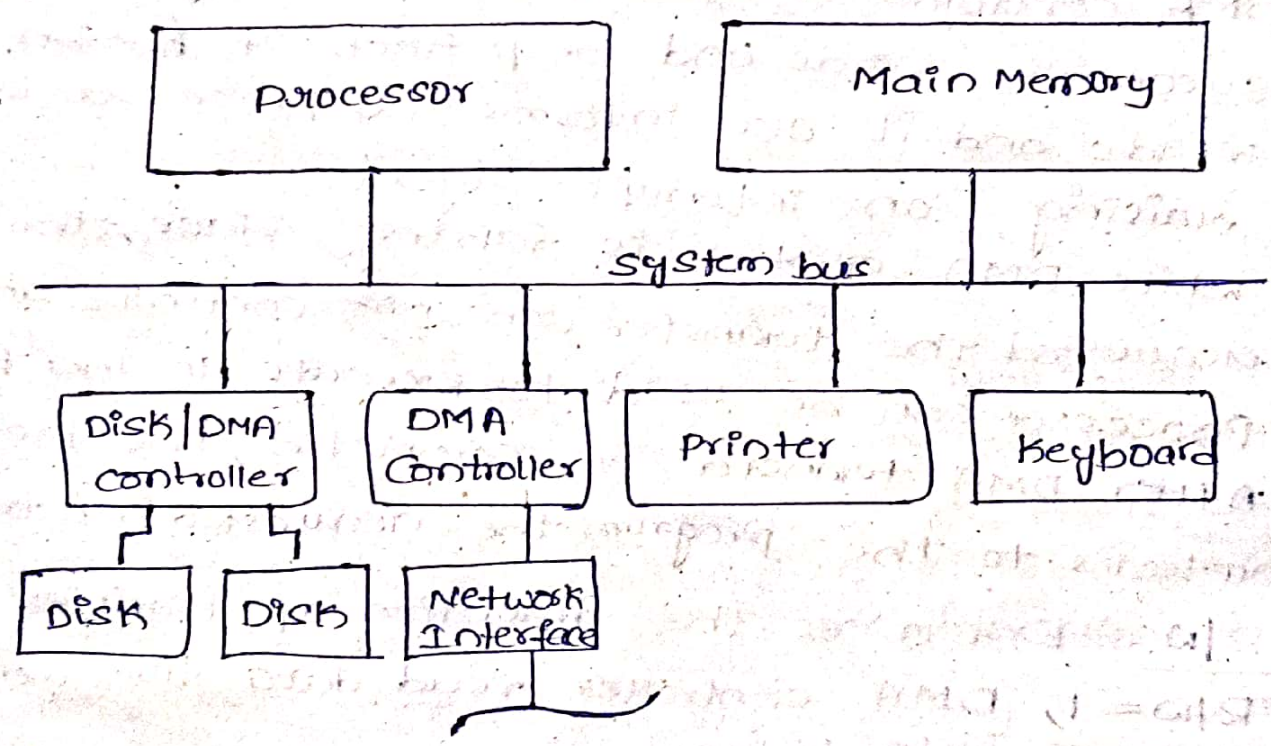


fig: Use of DMA controllers in a computer sys

A DMA controller connects a high speed network to the computer bus. The Disk controller two disks, also has DMA capability and it provides two DMA channels.

To start a DMA transfer of a block of data from main memory to one of the disks, the program

writes the address and the word count information into the registers of the corresponding channel of the disk controller.

When DMA transfer is completed, it will be recorded in status and control registers of the DMA channel. Done bit = IRQ = IE = 1.

Cycle Stealing :-

→ Requests by DMA devices for using the bus are having higher priority than processor requests.

→ Top priority is given to high speed peripherals such as, Disk, High speed Network Interface and Graphics display device.

→ Since the processor originates most memory access cycles, the DMA controller can be said to "steal" the memory cycles from the processor. This technique is called "cycle stealing"

Burst Mode: (or) Block Mode: (Entire block of data is transferred in one continuous sequence)

The DMA controller may be given Exclusive access to the main memory to transfer a block of data without interruption, This is known as Burst / Block Mode.

Bus Master:

The device that is allowed to initiate data transfers on the bus at any given time is called "Busmaster".

4.1: Bus Arbitration: [Sub topic in DMA]

It is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it.

Types: There are two approaches to bus arbitration on they are,

Centralized arbitaration: (A single bus arbiter performs arbitaration)

Distributed arbitaration: (All devices participate in the selection of next bus master)

Centralized Arbitaration

→ Here the processor is the bus master and it may grants bus mastership to one of its "DMA Controller"

→ A DMA controller indicates that it needs to become the bus master by activating the Bus-Request line (BR), which is an open drain line.

→ The signal on BR is the logical OR of the bus request from all devices connected to it

→ When BR is activated the processor activates the Bus Grant signal (BG1) and indicated the DMA controller that they may use the bus when it becomes free.

→ (BG1) This signal is connected to all devices using a daisy chain arrangement.

→ If DMA requests the bus, it blocks the propagation of Grant signal to other devices and it indicates to all devices that it is using the bus by activating open collector line, Busy (BBSY)

→ BBSY signal is 0, it indicates that the bus is busy. When BBSY becomes 1, the DMA controller which asserted BR can acquire control of the bus.

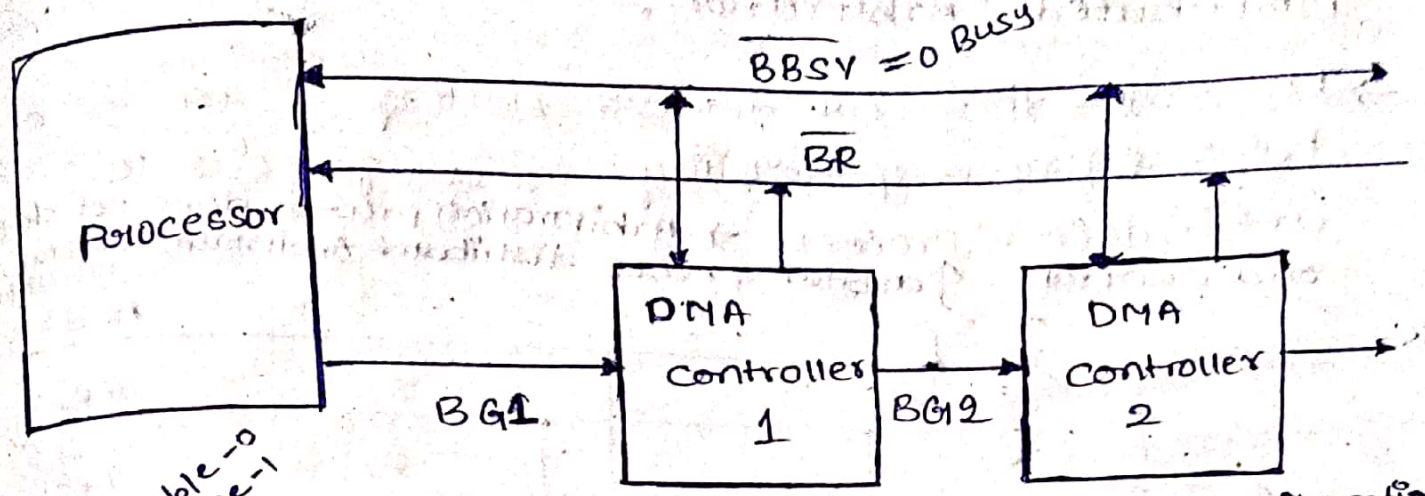


fig: A simple arrangement for bus arbitaration using a daisy chain

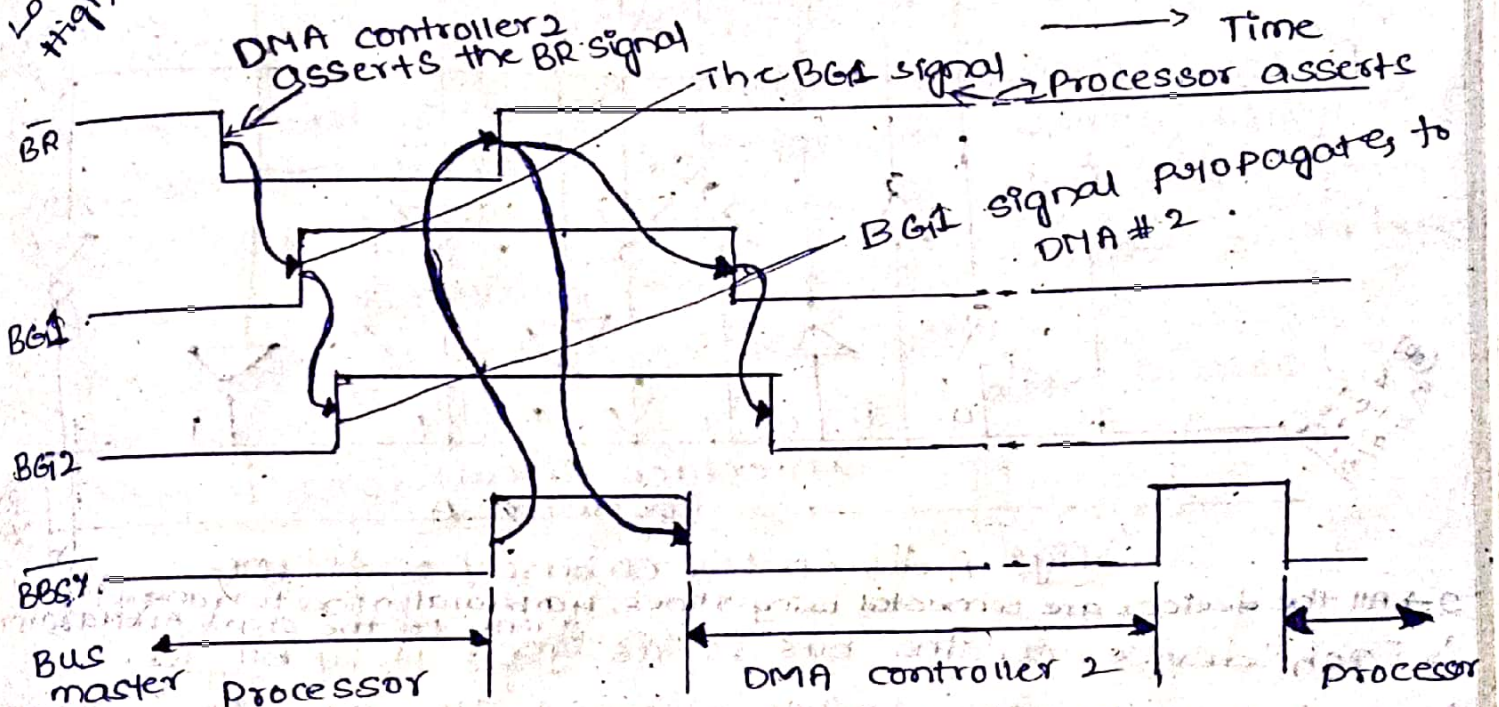


fig: sequence of signals during transfer of bus mastership for the devices in above fig.

The timing diagram shows the sequence of events for the devices connected to the processor is shown. DMA controller 2 requests and acquires bus mastership and later releases the bus. During its tenture as bus master, it may perform one or more data transfe After it releases the bus, the processor resources bus mastership.

Distributed Arbitration :-

1. It means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process. → Arbitaration proceds does not depend on a central arbitar & hence distributed Arbitaration has high reliability.

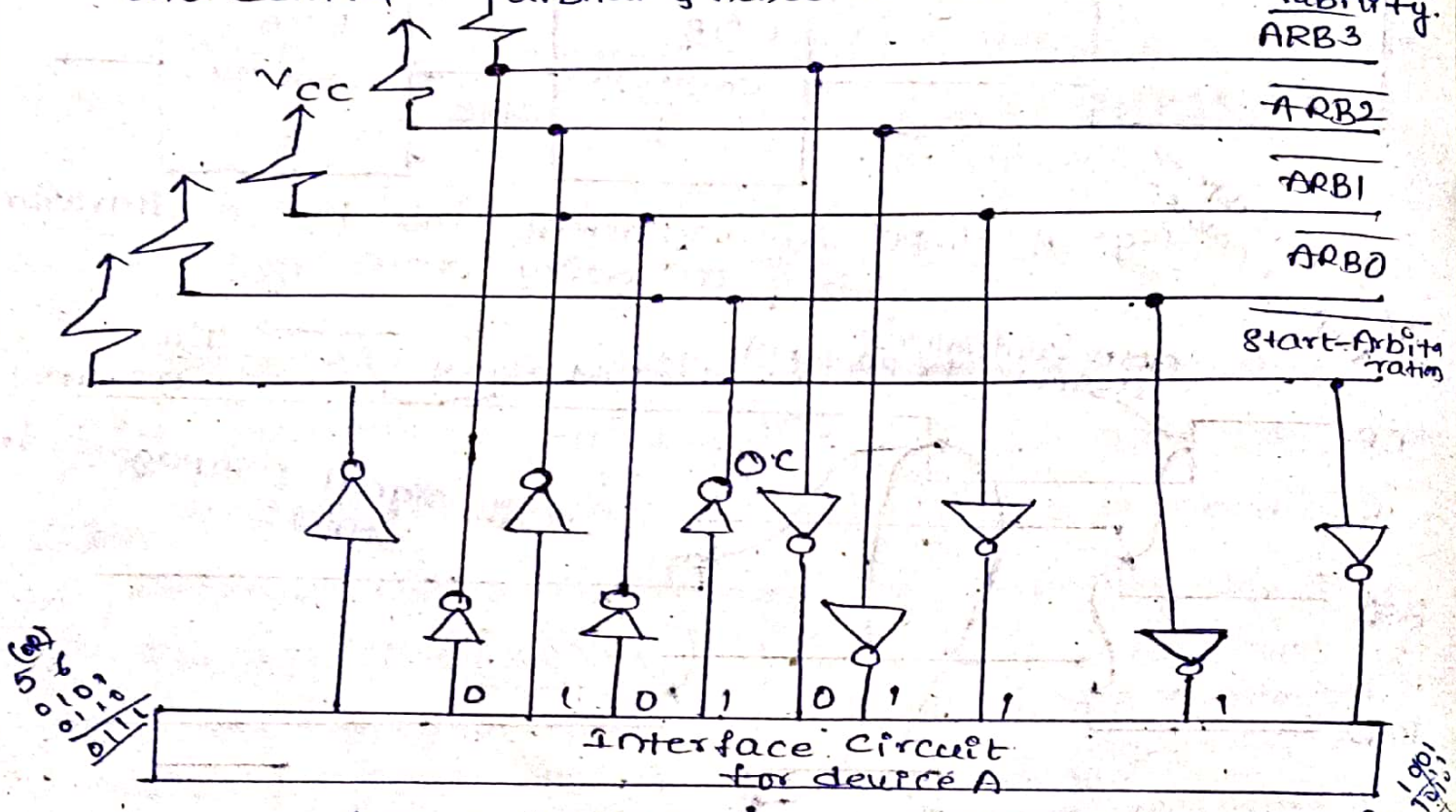


fig: A distributed arbitration scheme

2. → All the devices are connected using 5 lines, 4 arbitaration lines to transmit, a 4 bit ID & 1 line for the start Arbitaration
3. → Each device on the bus is assigned a 4 bit id.
4. → When one or more devices request the bus, they assert the start-Arbitration signal & place their 4-bit ID no. on four open collector lines, ARB0 to ARB3.
   → A winner is selected as a result of the interaction among the signals transmitted over these lines.
   → The net outcome is that the code on the four lines represents the request that has the highest ID number.
   → The drivers are of open collector type. Hence, if the i/p to one driver is Equal to 1, the i/p to another driver connected to the same bus line is equal to "0" (bus is in low-voltage state).

Topic 5: BUSES. → It a group of wires used to connect various devices.

→ A bus Protocol is the set of rules that govern the behavior of various devices connected to the bus.
→ The bus lines used for transferring data in to 3 types. They are,
- * Address line
- * Data line
- * control line

control signals :- specifies whether read/write operation has to be performed. The Bus control signals also carries timing information, they specify the time at which the processor & I/O devices place the data on the bus & receive the data from the bus.
→ During data transfer operation, one device plays the role of a "Master". Master device initiates the data transfer by issuing read/write command on the bus. Hence it is also called as "Initiator".
→ The device addressed by the master is called as Slave | Target.

Types of Buses :-
There are two types of buses, They are,
- * Synchronous Bus
- * Asynchronous Bus.

Timing :-

Timing refers to the way in which events are coordinated on the bus. Buses use either synchronous timing or asynchronous timing.

1. Synchronous bus :-

→ Includes a clock (→ mechanical or electrical device for measuring time.) in the control lines.
→ A fixed protocol for communication that is relative to the clock.
→ character is received at constant rate.

Advantage :- Involves very little logic and can run very fast.

Disadvantages :- Every device on the bus must run at the same clock rate and TO avoid clock skew (→ same source clock signal arrives at diff components at diff times), they can not be long if they are fast.

Synchronous Timing :-

→ The occurrence of events on the bus is determined by a "clock". The bus includes a clock line which transmits a regular sequence of alternating 1's and 0's of equal duration.
→ A single 1-0 transmission is a clock cycle (→ Amt of time blw 2 pulses) or bus cycle and defines a time slot.
→ All devices on the bus can read clock line, and all events start at the begining of a clock cycle.
→ Other bus signals may change the clock signal with slight delay.
→ Most events usually occupy a single clock cycle.

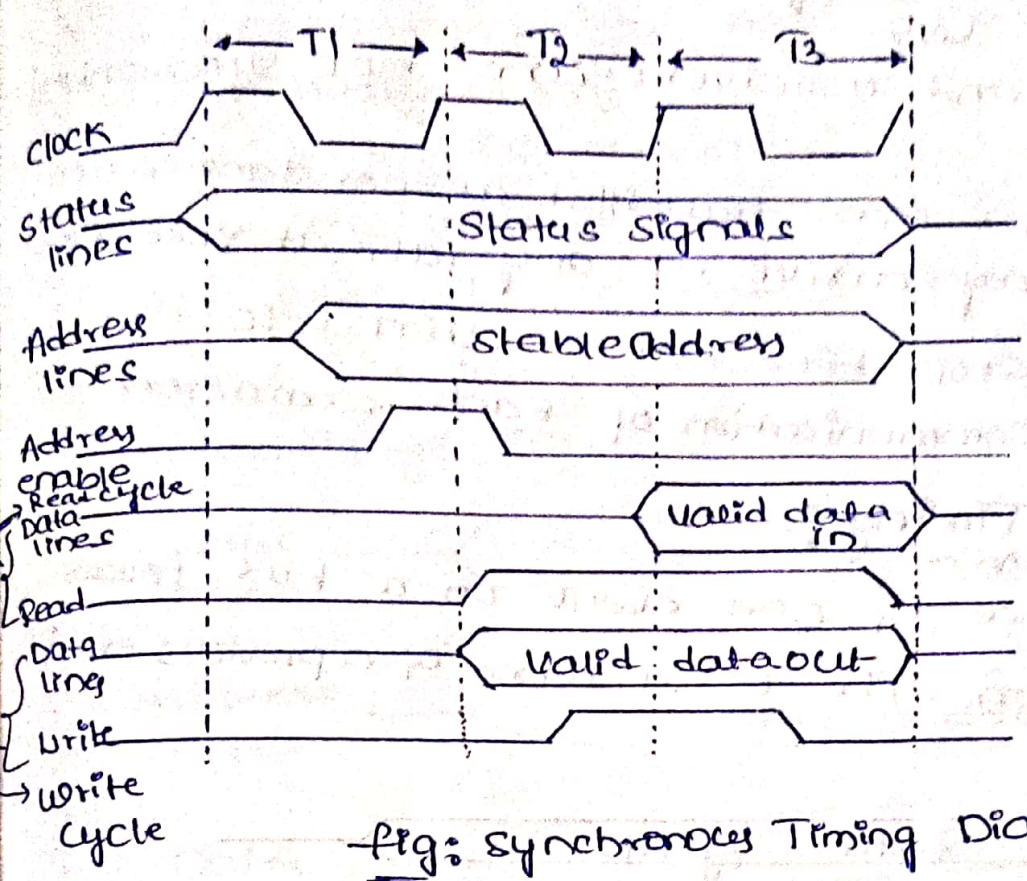


fig: Synchronous Timing Diagram

→ The processor places a memory address on the address line during the first clock cycle (T1), and may assert (declare) various status lines.

→ Once the address lines have stabilized (unique) the processor issues an address enable signal (T1).

→ For a read operation, the processor issues a read command at the start of the second cycle (T2).

→ A memory module recognizes the address and after a delay of one cycle, places the data on the datalines (T3).

→ For a write operation, the processor puts the data on the data lines at the start of the second cycle (T2) and issues a write command after the datalines have stabilized.

→ The memory module copies the i/f from the datalines during the third clock cycle.

2. Asynchronous Bus:-

→ Transmitters and receivers are not synchronised by clock.
→ Bit's of data are transmitted at constant rate.
→ character may arrive at any rate at receiver.
→ start and stop bits are required to establish communication of each character.

Asynchronous Timing:-

→ The occurrence of one event on a bus follows and depends on the occurrence of a previous event

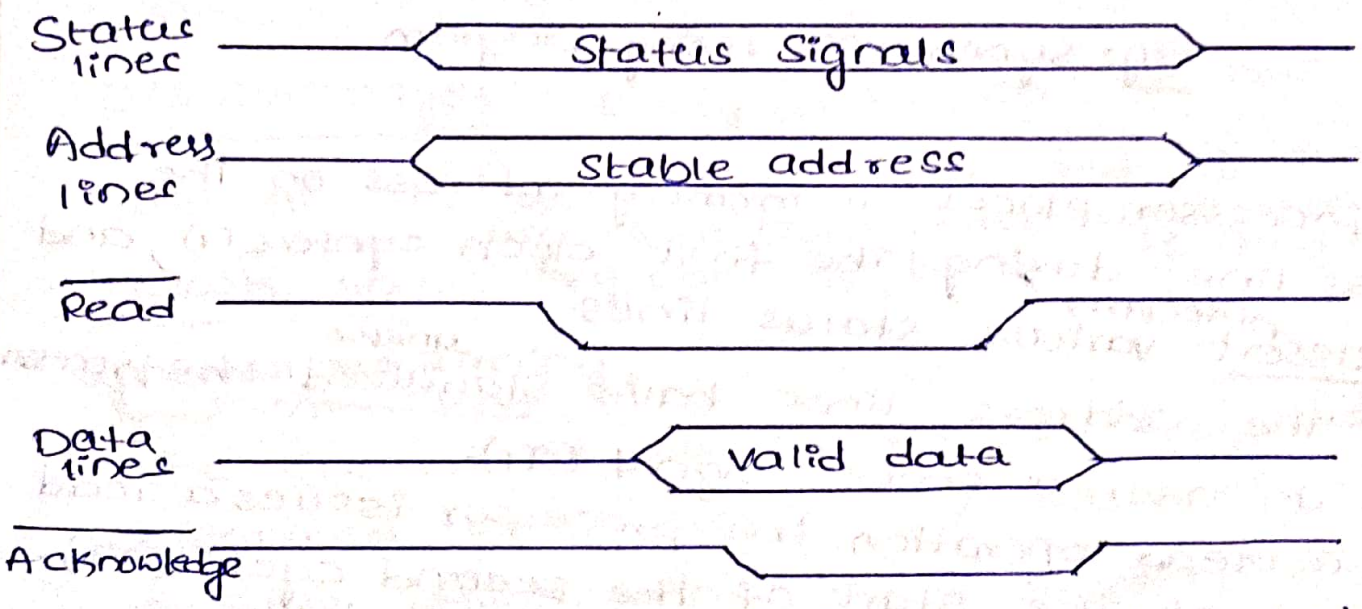


fig: Asynchronous Timing - system bus read cycle.

→ The processor places address and status signals on the bus.
→ After the signals have stablized the processor issues a read command, indicating the presence of valid address and control signals.
→ The appropriate memory decodes the address and responds by placing the data on the data line.

→ Once the data line has stabilized, the memory module asserts the read signal. This causes the acknowledged line to signal the processor that the data is available.

→ Once the master has read the data from the data lines, it de-asserts the read signal. This causes the memory module to drop the data and acknowledge lines.

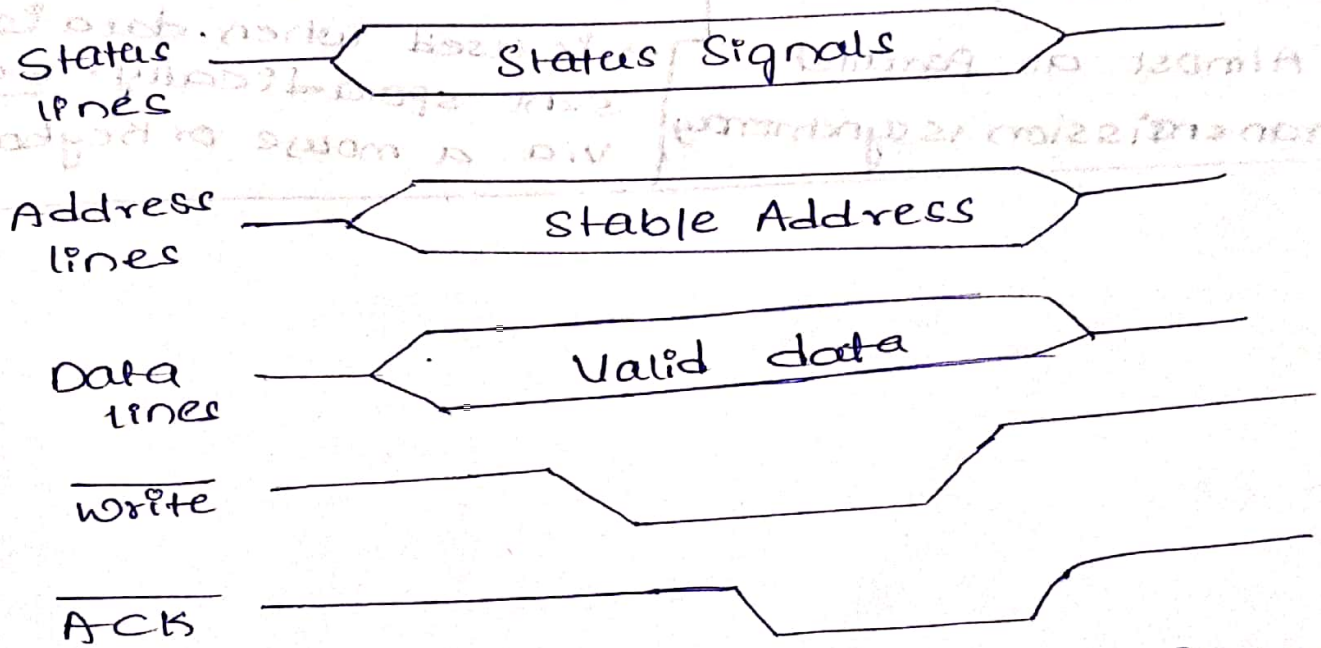


fig: Asynchronous Timing – system bus write cycle.

→ The master places the data on the data line at the same time that it puts signals on the status and address lines.

→ The memory module responds to the write command by copying the data from the data lines and then asserting the acknowledge line.

→ The master then drops the write signal and the memory module drops the ack signal.

→ Synchronous timing is simpler to implement & test, but with asynchronous timing, a mixture of slow & fast devices, using older & newer technology, can share a bus.

Synchronous	Asynchronous
→ Fast transmission	→ slower transmission due to the extra bits and the gaps.
→ Needs a common clock signal.	→ cheap & Easy to implement but no clock sharing.
→ May have to wait until data can be sent	→ can transmit when ready, no need to wait.
→ Almost all Parallel transmission is synchronous	→ is used when data is sent sporadically. Eg: via a mouse or keyboard.

TOPIC 6: Interface - Circuits.

→ I/O. interface consists of the circuitary required to connect an I/O device to a computer bus.

→ One side of the interface we have bus signals for: Address, Data, control Information.

→ Other side of the interface which connects to the I/O device has: Datapath and associated controls to transfer data between the interface and the I/O device. This side is called as a "port".

→ Ports can be classified in to two:
- Parallel port
- serial port

1. Parallel port

→ parallel port transfers data in the form of a number of bits, normally 8 or 16 to or from the device.

→ The connection between the device and the computer uses a multiple-pin connector and a cable with as many wires, typically arranged in a flat configuration.

→ The circuits at either end are relatively simple, as there is no need to convert between parallel and serial formats.

→ This arrangement is suitable for devices that are physically close to the computer.

In parallel port how a connection b/w keyboard to processor.

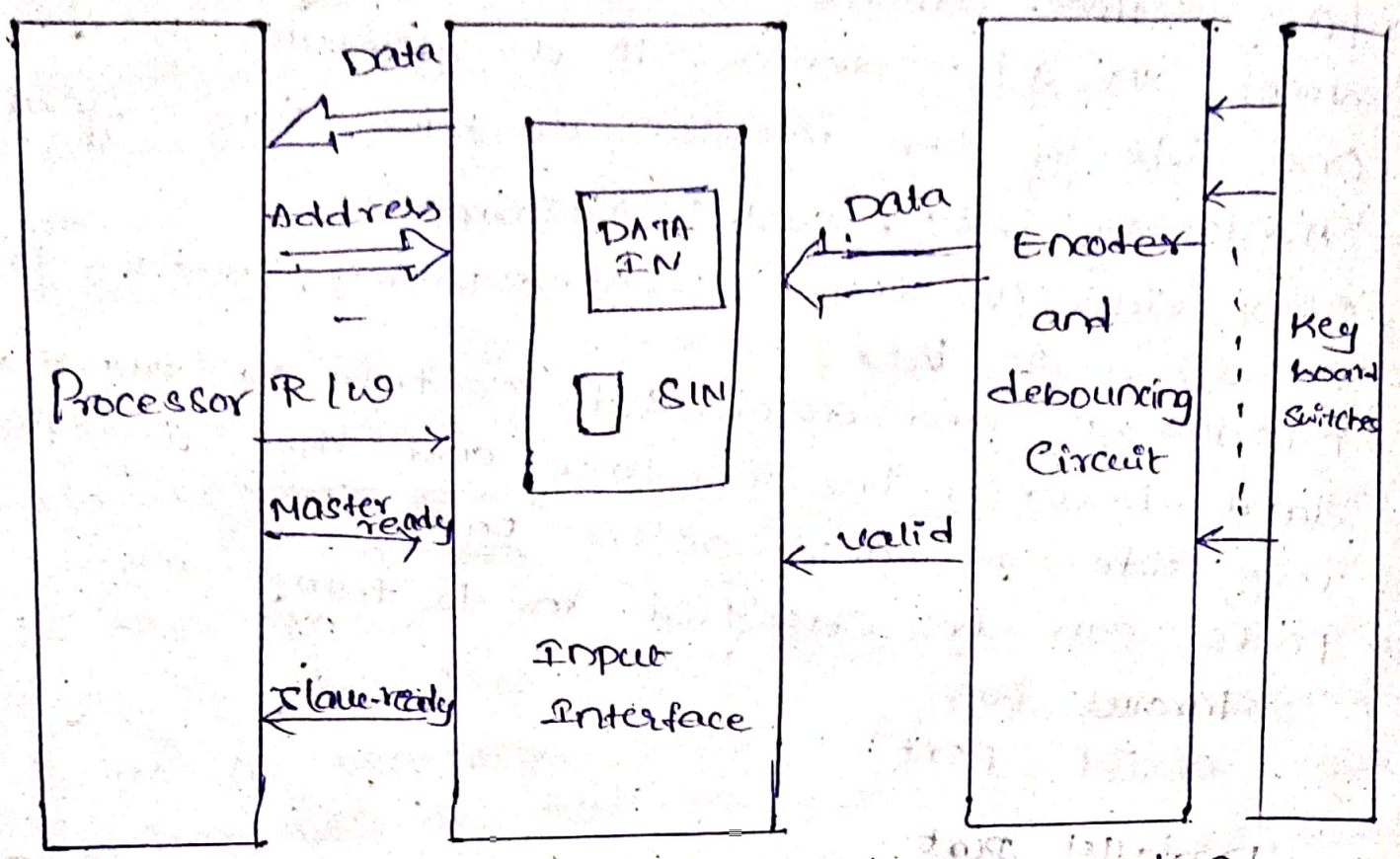


fig: Keyboard to Processor connection

→ Keyboard is connected to a processor using "parallel port"

* processor is 32-bits and uses memory-mapped I/O, and the asynchronous bus protocol.

* On the processor side of the interface we have:
  - Data lines
  - Address lines
  - control or R/W line
  - Master-ready signal and
  - slave-ready signal

* On the keyboard side of the Interface:
  - Encoder circuit which generates a code for the key pressed.
  - Debouncing circuit which eliminates the effect of a keybounce. (a single key stroke may appear as multiple events to a processor).
  - Data lines contain the code for the key.
  - valid lines changes from 0 to 1 when the key is pressed. This causes the code to be loaded into DATAIN & SIN to be set to 1.

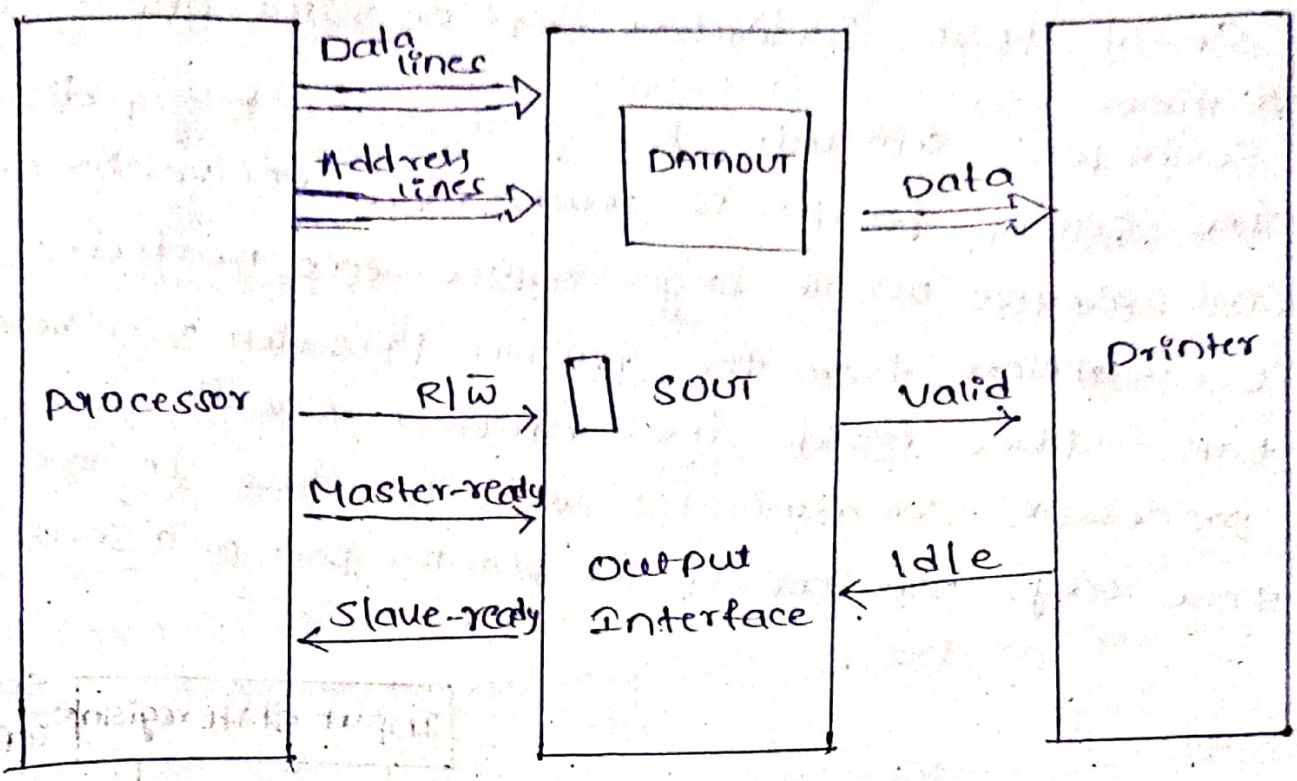


fig: printer to processor connection

→ Printer is connected to a processor using a "parallel port".
→ Processor is 32 bits, uses memory-mapped I/O and asynchronous bus protocol.
→ On the processor side:
- Data lines
- Address lines
- control or R/W line
- Master-ready signal and
- Slave-ready signal

→ On the printer side:
- Idle signal line which the printer asserts when it is ready to accept a character. This causes the SOUT flag to be set to 1.
- Processor places a new character into a DATAOUT register.
- valid signal, asserted by the interface circuit when it places a new character on the datalines.

2. Serial Port

→ Serial port transfers and receives one bit at a time.
→ Serial port suitable for devices at longer distance.
→ The serial format is much more convenient and cost effective where longer cables are needed.
→ Conversion from the parallel to serial vice versa takes place inside the interface circuit.
→ processor communicates with the bus in the same way, whether it is parallel port or a serial port.

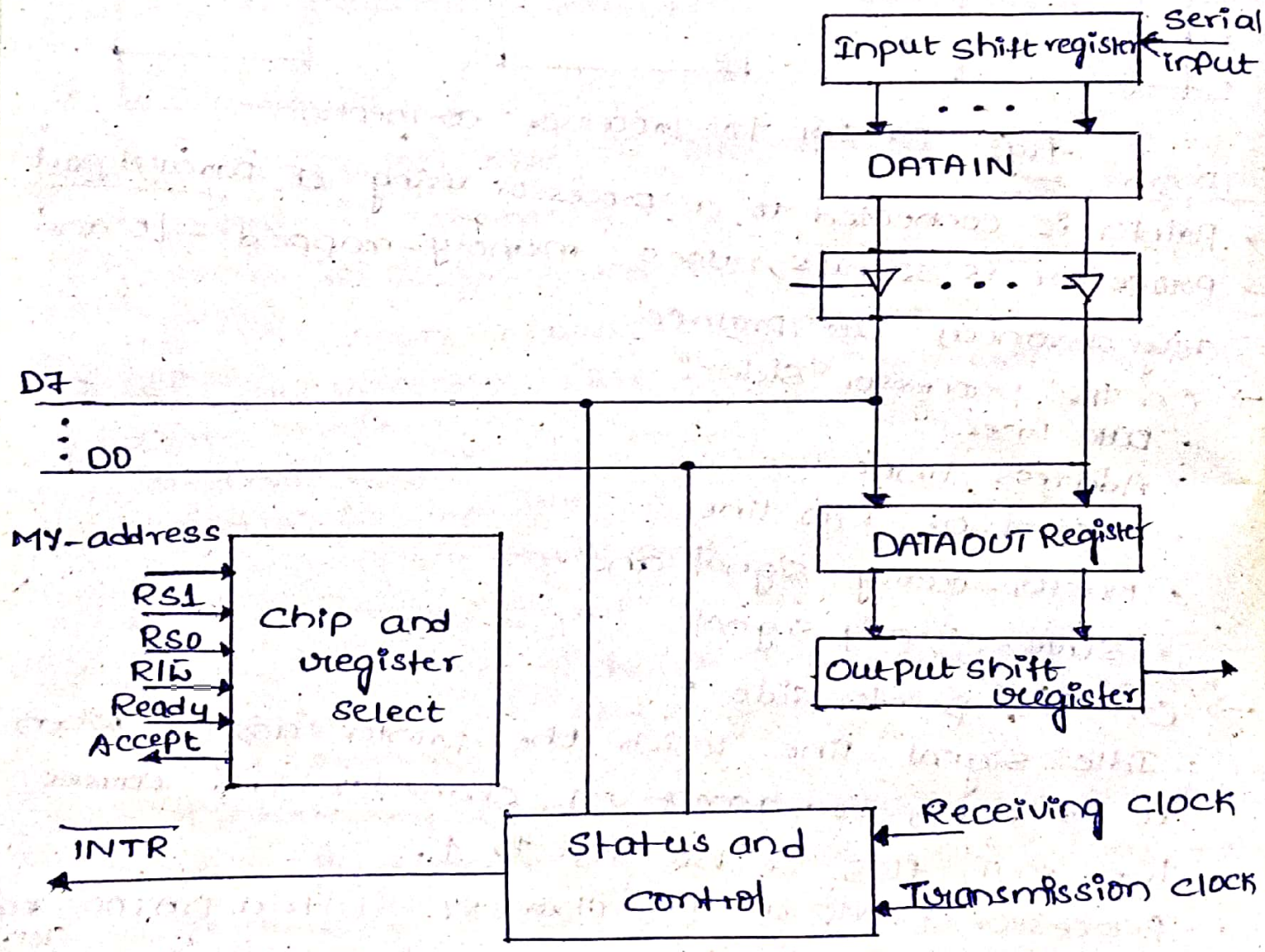


fig: A Serial Interface.

→ serial port is used to connect the processor to I/O devices that require transmission of data one bit a time.
→ serial port communicates in a bit-serial fashion on the device side and bit parallel-fashion on the bus side.
→ Transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.
→ Input shift register accepts input one bit at a time from the I/O device.
→ Once all the 8-bits are received, the contents of the input shift register are loaded in parallel into DATAIN register.
→ OUTPUT data in the DATAOUT register are loaded into output shift register.
→ Bits are shifted out of the output shift register and sent out to the I/O device one bit at a time.
→ As soon as data from the i/p shift register are loaded into DATAIN, it can start accepting another 8 bits of data.
→ Input shift register and DATAIN registers are both used at input so that the input shift register can start receiving another set of 8 bits from the input device after loading the contents to DATAIN, before the processor reads the contents of DATAIN. This is called as "double-buffering".

→ Serial interfaces require fewer wires, and hence serial transmission is convenient for connecting devices that are physically distant from the computer.

→ Speed of transmission of the data over a serial interface is known as the "bit rate"

→ Bit rate depends on the nature of the devices connected.

→ In order to accommodate devices with a range of speeds, a serial interface must be able to use a range of clock speeds.

→ Several standard serial interfaces have been developed:

→ Universal Asynchronous Receiver Transmitter (UART) for low-speed serial devices.

→ RS-232-C for connection to communication links.

Functions of interface circuits:-

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

TOPIC 7: Standard I/O Interfaces

→ I/O devices is connected to a comp using interface circuit.
→ There are several alternative designs for the bus of a computer.
Processor ↔ I/O
→ This variety means that I/O devices fitted with an interface circuit suitable for one computer may not be usable with other computers — a different interface may have to be designed for every combination of I/O device and computer, resulting many different Interfaces.
→ The most practical solution is to develop standard interface signals and protocols.
→ A personal computer has:
- A motherboard which houses the processor chip, main memory and some I/O interfaces.
- Also has a few connectors into which additional interfaces can be plugged.

→ processor bus is defined by the signals on the processor chip.
→ • Devices which require high-speed connection to the processor are connected directly to the bus.
→ Because of electrical reasons only a few devices can be connected directly to the processor bus.
→ → Motherboard usually provides another bus that can support more devices.
→ • processor bus and the other bus are interconnected by a circuit called "Bridge".
→ • Devices connected to the expansion bus experience a small delay in data transfers.
→ Design of Processor bus is closely tied to the architecture of the processor.
• No Uniform standard can be defined.

Three widely used bus standards:

1. PCI (Peripheral Component Interconnect)
2. SCSI (small computer system Interface)
3. USB (universal serial Bus)

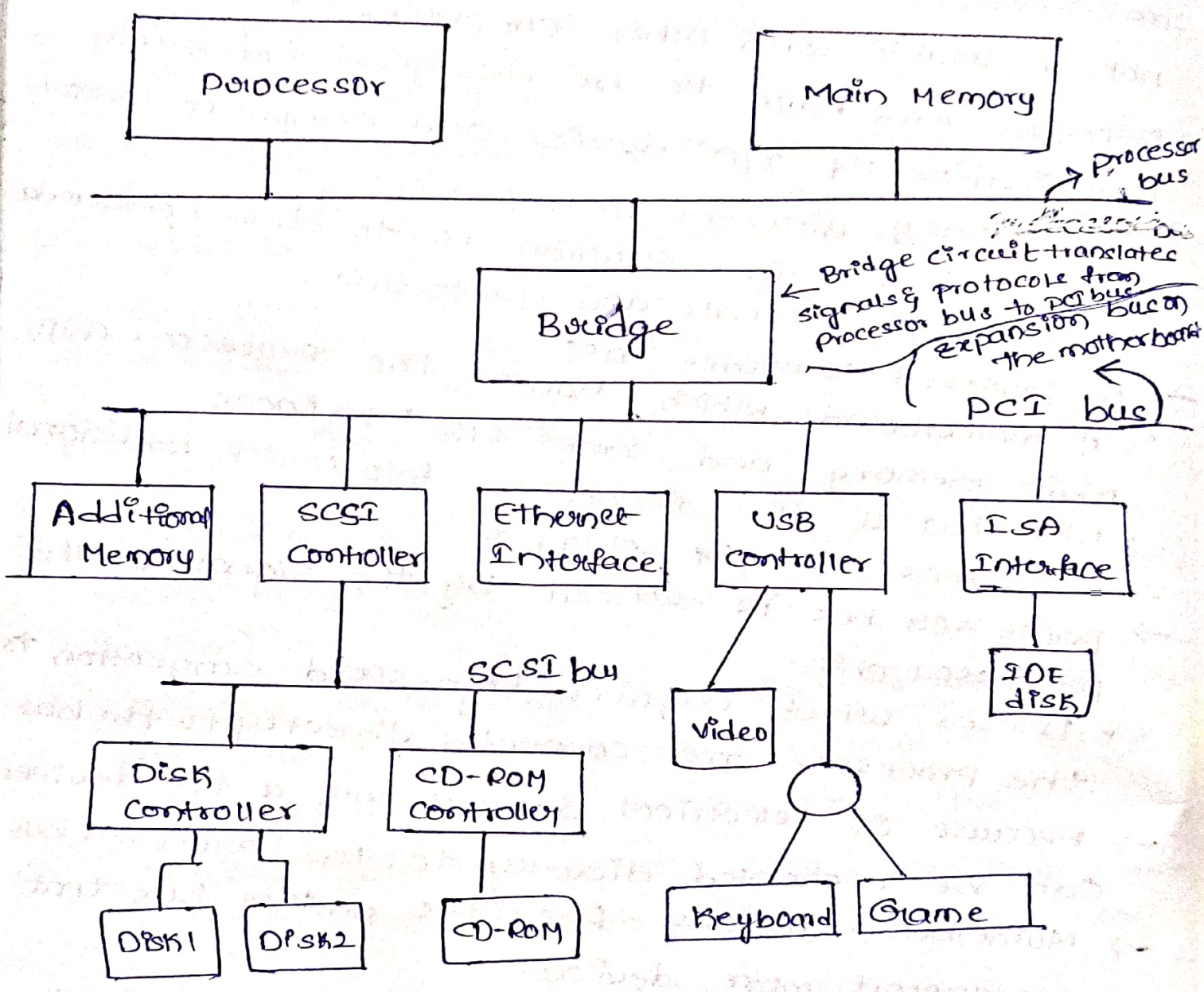


fig: An example of a computer system using different Interface standards.

→ PCI defines an expansion bus on the motherboard.
→ SCSI and USB are used for connecting additional devices both inside and outside the computer box.
→ The SCSI bus is high-speed parallel bus intended for devices such as disks and video displays.

→ USB buses uses serial transmission to suit the needs of equipment ranging from keyboards to game controls to internet connections.

1. PCI (Peripheral Component Interconnect)

→ Low-cost bus, processor independent, plug-and-play capability. (to connect a new device, the user simply connects the device interface board to the bus).

→ In today's computers, most memory transfers involve a burst of data rather than just one word. The PCI is designed primarily to support this mode of operation.

→ The bus supports three independent address spaces: memory, I/O, and configuration.

→ We assumed that the master maintains the address ilf on the bus until data transfer is completed. But, the address is needed only long enough for the slave to be selected. Thus the address is needed on the bus for one clock cycle only, freeing the address lines to be used for sending data in subsequent clock cycles. The result is a significant cost reduction.

→ A master is called the initiator in PCI terminology. The addressed device that responds to read and write commands is called a "target".

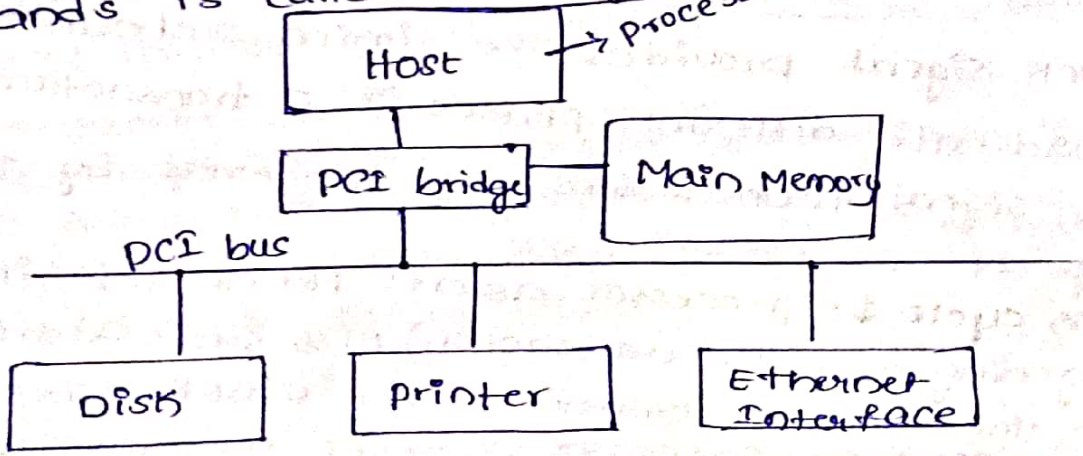


fig: Use of a PCI bus in a computer system.

Data transfer signals on the PCI bus

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

→ consider a bus transaction - processor (initiator) read four 32-bit words from memory (target).

→ A complete transfer operation on the bus, involving an address and a burst of data is called a "transaction"

→ Individual word transfer within a transaction are called a "phases"

→ Clock signal provides the timing reference used to coordinate different phases of a transaction.

→ All signal transactions are triggered by the rising edge of the clock.

Clock cycle 1: processor asserts FRAME# to indicate the beginning of a txn (transaction) Also sends address on the AD lines and a command on the C/BE# lines.

Clock cycle 2: processor removes the address & disconnects its drivers from the AD lines.

2. SCSI Bus:- (Small computer system Interface):

SCSI refers to the standard bus which is defined by ANSI (American National Standard Institute).

SCSI bus the several options it may have be

Narrow bus → It has 8 data lines & transfers 1 byte at a time

Wide bus → It has 16 data lines & transfers 2 byte at a time

Single-Ended Transmission → signal uses separate wire

HVD (High Voltage Differential) → It uses 5V (TTL cells)

LVD (Low voltage Differential) → It uses 3.3 V.

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

→ The transfer rate depends on, Length of the cable. Number of devices connected.

→ To acheive high transfer rate, the bus length should be 1.6 m for single Ended SE signaling and 12 m for LVD signaling.

→ The SCSI bus connected to the processor through the SCSI controller. The data are stored on a disk in blocks called "Sectors".

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

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

- Initiator
- Target

Initiator:- It has the ability to select a particular target & to send commands specifying the operation to be performed.
They are the controllers on the processor side.

Target:
The disk controller operates as a target
It carries out the commands it receive from the initiator. The initiator establishes a logical connection with the intended target.

SCSI bus signals:

Category	Name	Function
1. Data	- DB(0) to DB(7) - DB(P)	Datalines parity bit for databus
2. Phases	- BSY - SEL	Busy Selection
3. Information type	- C/D - MSG	Control | Data Message
4. Handshake	- REQ - ACK	Request Acknowledge
5. Direction of Transfer	- I/O	Input/output
6. Other	- ATN - RST	Attention Reset

→ All signal names are proceeded by minus sign.

Phases in SCSI Bus:-
The phases in SCSI bus operation are,
* Arbitaration
* selection
* Information transfer
* Reselection

3. Universal serial Bus (USB)

→ A modern computer system is likely to involve a wide variety of devices such as keyboards, microphones, cameras, speakers and display devices.
→ Most computers also have a wired or wireless connection to the Internet.
→ A key requirement of such an environment is the availability of simple, low-cost mechanism to connect these devices to the computer.
→ An important recent development in this regard is the USB.

Key objectives of USB :-

→ provide simple, low cost and easy to use interconnection system that overcomes the difficulties due to limited number of I/O ports available on a computer.
→ Accommodate wide range of data transfer characteristics for I/O devices like variation in speed, volume and timing constraints associated with data transfers. Enhance user convenience through a "plug-and-play" mode of operation.

Ex: An additional speaker can be connected at any time while system is operating - The system should detect the existence of this new device automatically, identifies the device driver software and any other facilities needed to serve that device and establish the appropriate address and logical connections to enable them to communicate.

Hence USB makes it possible to connect many devices to computer system at any time.

USB Architecture:-

→ Serial transmission format is chosen for USB - low cost and flexibility requirement.

→ clock and data i/f are coded together and transmitted as a single signal.

→ Hence there are no limitations on clock frequency or skew problem (same source clock signal arrives diff comp at diff time) and provide high data transfer bandwidth by using a high clock frequency.

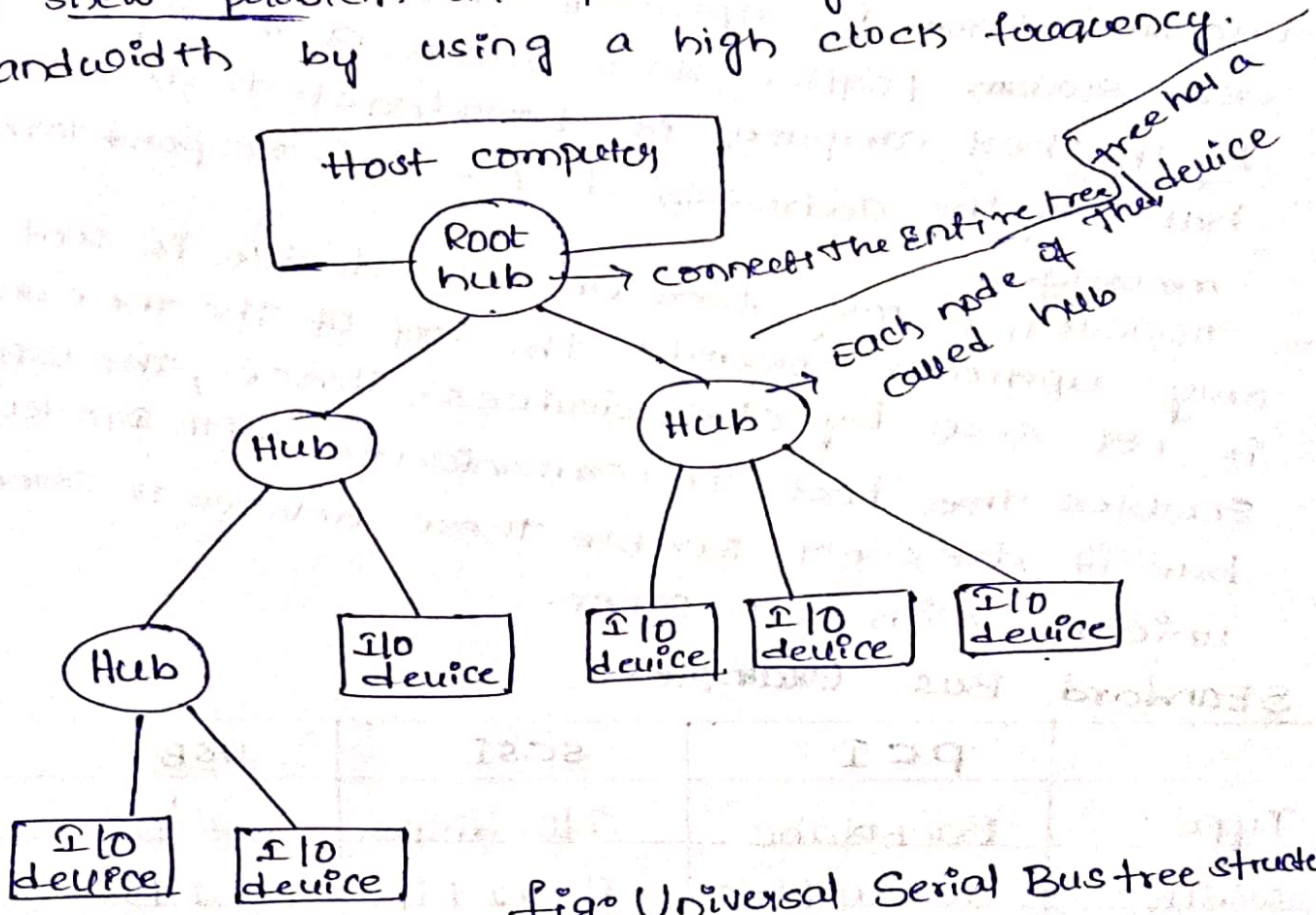


fig: Universal Serial Bus tree structure

→ A device may send a message only in response to a poll message from the host, hence upstream msg do not encounter conflicts with each other.

→ To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure.

Each node of the tree has a device called a "hub", which acts as an intermediate control point between the host and 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.

Ex:- keyboard, Internet connection, speaker or digital cam.

→ 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 all I/O devices, but only the addressed device will respond to that message.

→ However, a msg 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 I/O devices, but it does not enable these devices to communicate with each other.

Standard Bus Examples:

	PCI	SCSI	USB
Type	Backplane	I/O-disks	I/O
width	32-64 bits	8-32 bits	1 bit
Multiplexed?	Yes	Yes	Yes
Clocking	33 or(66) MHz	5(10)MHz	Asynchronously
Data rate	133 (266) MB/s	10(20) MB/s	0.2, 1.5, 80MB/s
Arbitration	Parallel	self selection	Daisy-chain
Min Masters	1024	7-31	127
Min length	0.5 m	2.5m	