MODULE - 5

INPUT / OUTPUT ORGANIZATION

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:

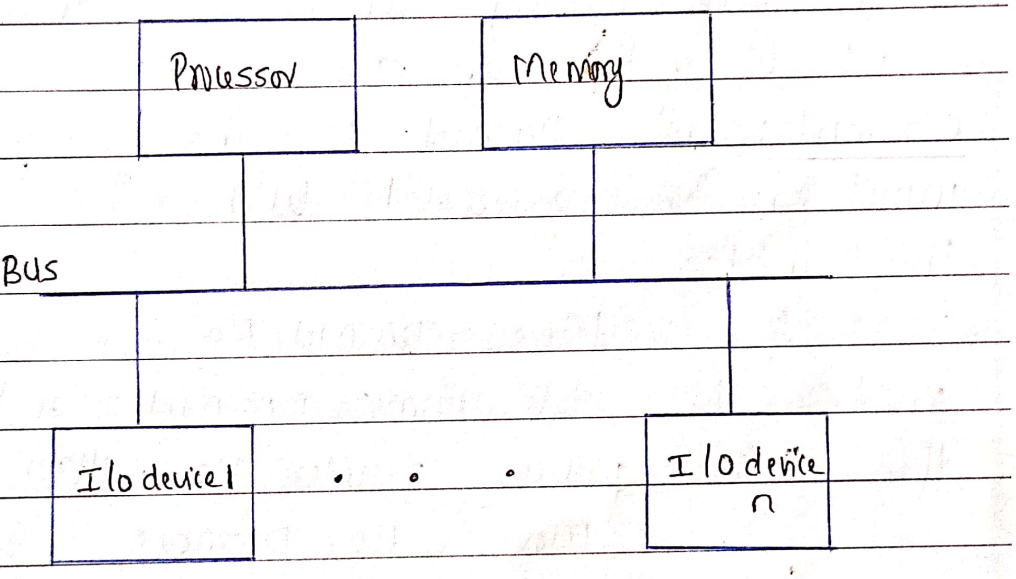


fig: 4.1 A single-bus structure

↳ the bus enables all the devices connected to it to exchange information.

↳ It consists of three sets of lines used to carry address, data and control signals.

↳ Each I/o device is assigned to a unique set of addresses.

↳ When the processor places a particular address on the address lines, 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 transfered over data lines.

↳ When I/O devices and mly share the same address space, the arrangement is called memory mapped I/O.

↳ With memory mapped I/O, any m/c inst: that can access memory can be used for transferring data to or from an I/O device.

Example: if DATAIN is the address of the input buffer associated with the keyboard, the inst:

Move DATAIN, R0

reads the data from DATAIN and stores them into processor register R0. Similarly the inst:

Move R0, DATAOUT

sends the contents of R0 to location DATAOUT which may be the o/p buffer of a display unit or a printer.

I/O interface for an input device

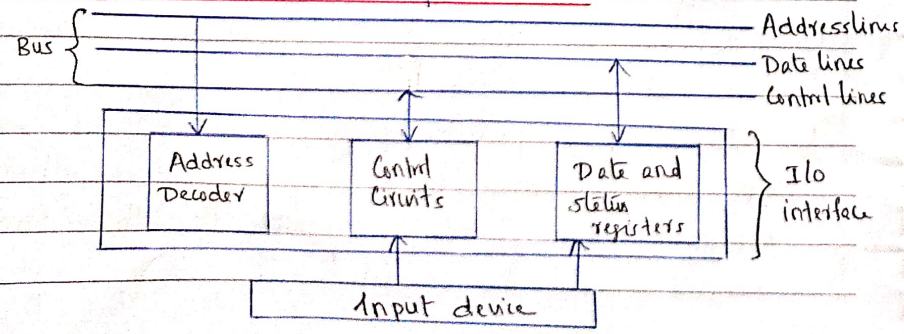


fig: shows the hlw required to connect an I/O device to bus.

↳ The address decoder enables the device to recognize its address when this address appears on the address lines.

↳ The data register holds the data being transferred to or from the processor.

↳ The status register contains the info: relevant to the operation of I/O device.

↳ Both the data and status registers are connected to the data bus and assigned unique address

↳ The address decoder, the data and status registers and the control circuitry required to coordinate I/O transfers constitute the device's interface circuit.

INTERRUPTS

↳ There are many situations where other tasks can be performed while waiting for an I/O device to become ready.

↳ To allow this to happen, we can arrange for the I/O device to alert the processor when it becomes ready.

↳ It can do so by sending a hlw signal called an interrupt to the processor.

↳ At least one of the bus control lines, called an

interrupt-request line, is usually dedicated for this purpose.

↳ Since the processor is no longer required to continuously check the status of external devices, it can use the waiting period to perform other useful functions.

↳ By using interrupts, such waiting periods can be ideally be eliminated.

EXAMPLE : Let program consist of two routines COMPUTE and PRINT.

Assume that COMPUTE produces a set of n lines of o/p, to be printed by the PRINT routine

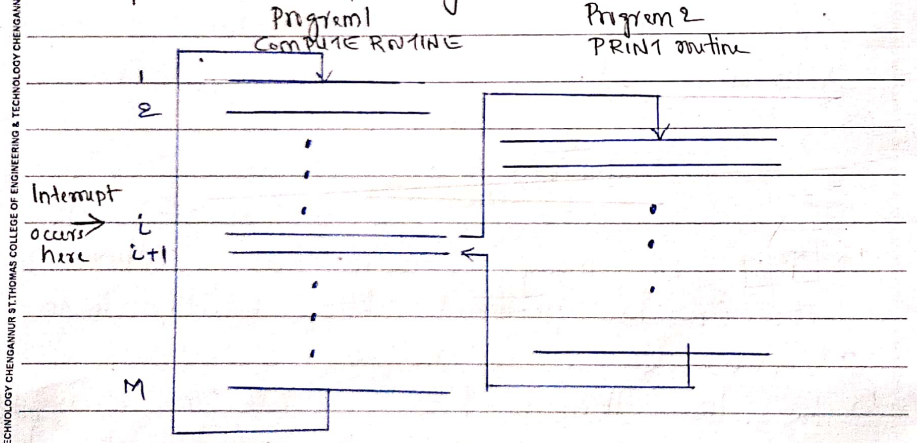


fig: 4.5 Transfer of control through the use of interrupts.

The routine executed in response to an interrupt-request is called the interrupt-service routine,

which is the PRINT routine in our example.

↳ Assume that an interrupt request arrives during execution of inst: i in fig: 4.5

↳ the CPU first completes execution of inst: i

↳ then it loads the PC with the address of the first inst: of the ISR.

↳ After execution of the ISR, the processor has to come back to inst: i+1.

↳ ∴ when an interrupt occurs, the current contents of the PC, which points to inst: i+1, must be put in temporary storage in a known location.

↳ A return-from-interrupt inst: at the end of the interrupt-service routine reloads the PC from that temporary storage location, causing execution to resume at inst: i+1.

↳ In many processors, the return address is saved on the processor stack.

↳ The processor must inform the device that its request has been recognized so that it may remove its interrupt-request signal. This may be accomplished by means of a special control signal on the bus called interrupt-acknowledge signal.

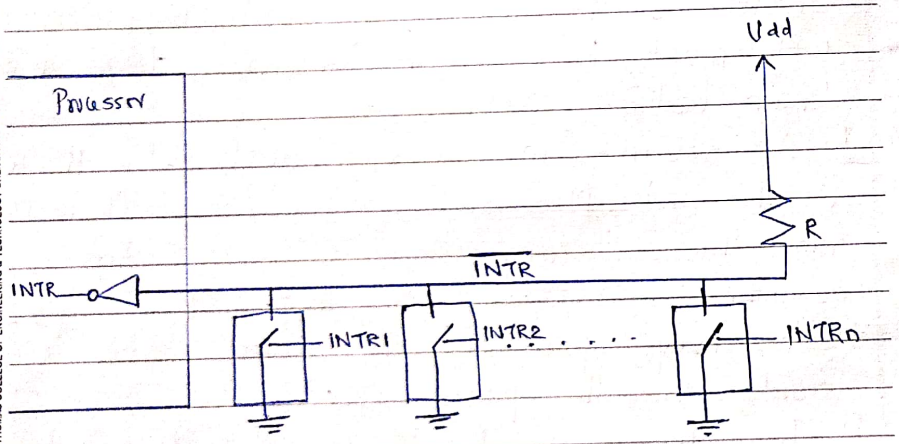
↳ The delay between the time an interrupt request is received and start of execution of the

ISR is called interrupt latency.

INTERRUPT HARDWARE

↳ An I/O device requests an interrupt by activating a bus line called interrupt-request. Most computers are likely to have several I/O devices that can request an interrupt.

↳ A single interrupt-request line may be used to serve n devices as shown in fig:



↳ All the devices are connected 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.

↳ When a device requests an interrupt by closing its switch, the voltage on the line drops to 0, causing the interrupt-request signal INTR, received by the processor to go to 1.

↳ 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 device, that is

$$INTR = INTR_1 + \ldots + INTR_n$$

↳ It is customary to use $\overline{INTR}$, to name the interrupt-request signal on common line, because this signal is active when in the low-voltage state.

↳ Register R is called a pull-up register because it pulls the line voltage up to the high voltage state when the switches are open.

ENABLING AND DISABLING INTERRUPTS

↳ The arrival of an interrupt request from an external device cause the processor to suspend the execution of one program and start the execution of another. Because interrupts can arrive at any time, they may alter the sequence of events from that envisaged 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 and disable such interruptions as desired.

↳ The first possibility is to have the processor hardware ignore the interrupt-request line until the execution of the first instruction of ISR has been completed.

↳ Then, by using an Interrupt-disable instruction as the 1st instruction in the ISR, the programmer can ensure that no further interruptions will occur until an Interrupt-enable instruction is executed. Interrupt-enable instruction will be the last instruction in the ISR before the Return-from-interrupt instruction.

↳ The second option, which is suitable for a simple processor with only one interrupt-request line is to have the processor automatically disable interrupts before starting the execution of the ISR. After saving the contents of the PC and the processor status register (PS) on the stack, the processor performs the equivalent of executing an Interrupt-disable instruction. One bit in the PS register, called Interrupt-enable, indicates whether interrupts

are enabled. An interrupt request received while this bit is equal to 1 will be accepted. After saving the contents of the PS on the stack, with the Interrupt-enable bit equal to 1, the processor clears the Interrupt-enable bit in its PS register, thus disabling further interrupts. When a Return-from-interrupt instruction is executed, the contents of the PS are restored from the stack, setting the Interrupt-enable bit back to 1. Hence interrupts are again enabled.

↳ In the third option, the processor has a special interrupt-request line for which the interrupt-handling circuit responds only to the leading edge of the signal. Such a line is said to be edge-triggered. In this case, the processor will receive only one request, regardless of how long line is activated.

Sequence of events involved in handling an interrupt request from a single device.

Assuming that interrupts are enabled, the following is a typical scenario:

1. The device raises an interrupt request.
2. The processor interrupts the program currently being executed.
3. Interrupts are disabled by changing the control

bits in the PS (except in the case of edge-triggered interrupts).

4. The device is informed that its request has been recognized, and in response, it deactivates the interrupt request sig.
5. The action requested by the interrupt is performed by ISR.
6. Interrupts are enabled and execution of the interrupted program is resumed.

VECTORED INTERRUPTS

A device requesting an interrupt can identify itself by sending a special code to the processor over the bus. This enables the processor to identify individual devices even if they share a single interrupt-request line. The code supplied by the device may represent starting address of the interrupt-service routine for that device. The code length is typically in range of 4 to 8 bits. The remainder of the address is supplied by the processor based on the area in its memory where the address of the interrupt service routines are located.

↳ The location pointed to by the interrupting device is used to store the starting address of the interrupt-service routine. The processor reads this address

called the interrupt vector, and loads it into PC. The interrupt vector may also include a new value for the processor status register.

INTERRUPT NESTING

For some devices, a long delay in responding to an interrupt request may lead to erroneous operations.

↳ Consider for eg:, a computer that keep track of the time of day using a real-time clock. This is a device that sends interrupt requests to the processor at regular intervals. For each of these request, the processor executes a short interrupt-service routine to increment a set of counters in the memory that keep track of time in seconds, minutes and so on. Proper operation requires that the delay in responding to an interrupt request from the real time clock be small in comparison with the interval between two successive requests. To ensure that this requirement is satisfied in the presence of other interrupting devices, it may be necessary to accept an interrupt request from the clock during the execution of an interrupt-service routine for another device.

↳ This example suggests that I/O devices should be organized in a priority structure. An interrupt request from a high-priority device should be accepted

while the processor is serving another request from a lower-priority device.

↳ A multiple-level priority organization means that during execution of an interrupt-service routine, interrupt requests will be accepted from some devices but not from others, depending upon the device's priority. To implement this scheme, we can assign a priority level to the processor that can be changed under program control. The priority level of the processor is the priority of the program that is currently being executed. The processor accepts interrupts only from devices that have priorities higher than its own.

↳ The processor's priority is usually encoded in a few bits of the processor status word. It can be changed by program instructions that write into the PS. These are privileged instructions, which can be executed only while the processor is running in the supervisor mode. The processor is in the supervisor mode only when executing operating system routines. It switches to the user mode before beginning to execute application programs. Thus, a user program cannot accidentally or intentionally, change the priority of the processor and disrupt the system's operation. An attempt to execute a privileged instruction while in the user mode leads to a special

type of interrupt called a privilege exception.

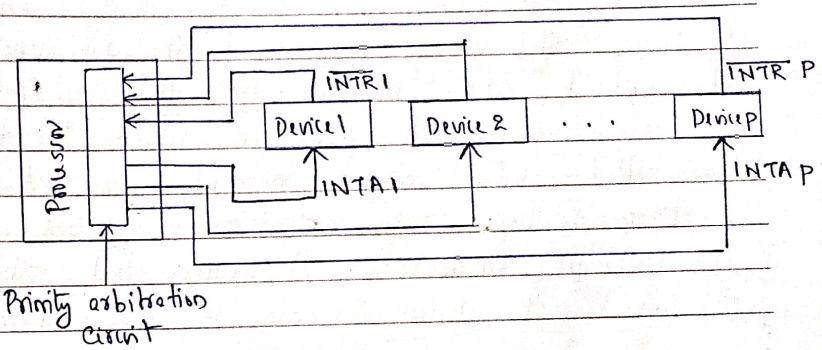


Fig 4.7 Implementation of interrupt priority using individual interrupt-request and acknowledge lines.

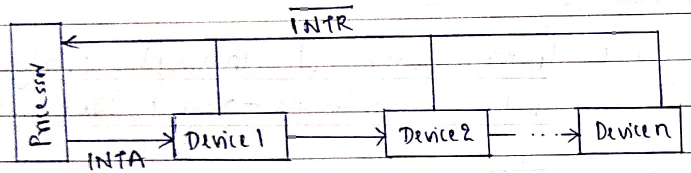
A multiple-priority scheme can be implemented easily by using separate interrupt request and interrupt-acknowledge lines for each device as shown in fig. Each of the interrupt-request lines is assigned a different priority level. Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor. A request is accepted only if it has a higher priority level than that currently assigned to the processor.

SIMULTANEOUS REQUESTS

Let's now consider the problem of simultaneous arrivals of interrupt requests from two or more devices

The processor must have some means of deciding which request to service first.

5 Polling the status registers of the I/O devices is the simplest such mechanism. In this case priority is determined by the order in which the devices are polled. When vectored interrupts are used, we must ensure that only one device is selected to send its interrupt vector code. A widely used scheme is to connect the devices to form a daisy chain

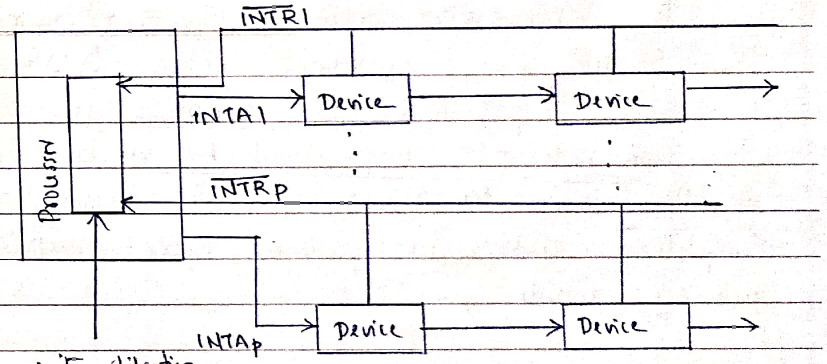


4.8 (a) Daisy chain

The interrupt-request line $\overline{INTR}$ is common to all devices. The interrupt-acknowledge line INTA is connected in a daisy chain fashion, such that the INTA signal propagates serially through the devices. When several devices raise an interrupt request and the $\overline{INTR}$ line is activated, the processor responds by setting the INTA line to 1. This signal is received by device 1. Device 1 passes the signal on to device 2 only if it doesn't require any service. If device 1 has a pending request for interrupt, it blocks the INTA signal and proceeds to put its identifying

Code on the data lines. Therefore, in the daisy-chain arrangement, the device that is electrically closest to the processor has the highest priority. The second device along the chain has second highest priority.

The two schemes 4.7 and 4.8a are combined to produce the more general structure in fig: 4.8b.



4.8 (b) Arrangement of priority groups

Devices are organized in groups and each group is connected at a different priority level. Within a group, devices are connected in a daisy chain. The organization is used in many computer systems.

EXCEPTIONS

The term exception is often used to refer to any event that causes an interruption.
eg: I/O interrupts.

RECOVERY FROM ERRORS

When exception processing is initiated as a result of such errors, the processor proceeds in exactly in the same manner as in the case of an I/O interrupt request. It suspends the program being executed and starts an exception-service routine. This routine takes appropriate action to recover from the error, if possible or to inform user about it. When an interrupt is caused by an error / execution of the interrupted instr cannot usually be completed, and the processor begins exception processing immediately.

DEBUGGING

- System s/w usually includes a program called a debugger, which helps the programmer to find errors in a program.
- The debugger uses exceptions to provide two important facilities called trace and breakpoints.
- When a processor is operating in the trace mode, an exception occurs after execution of every instr, using the debugging program as the exception-service routine.
- The debugging program enables the user to examine the contents of registers, memory locations and so on. The trace exception is disabled during the execution of the debugging program.
- Breakpoints provide a similar facility, except that the program being debugged is interrupted only at specific points selected by the user.
- An instr called Trap or Software-interrupt is usually provided for this purpose.

Privilege Exception

To protect the operating system of a computer from being corrupted by user programs, certain instrs can be executed only while the processor is in the supervisor mode. They are called privileged instrs.
for eg: When the processor is running in the user mode, it will not execute an instr that changes the priority level of the processor or that enables a user program to access areas in the computer memory that have been allocated to other users. An attempt to execute such an instr will produce a privilege exception.

DIRECT MEMORY ACCESS

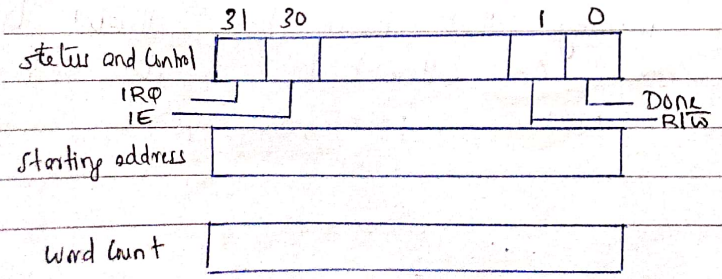
↳ The transferring of large block of data directly between an external device and the main memory without continuous intervention by the processor is called DMA.

↳ DMA transfers are performed by a control circuit that is part of the I/O device interface. This control circuit is called DMA Controller.

↳ The DMA Controller performs the functions that would normally be carried out by the processor when accessing the main mly.

↳ To initiate the transfer of a block of words, the processor sends the starting address, the no: of words in the block, and the direction of the transfer. On receiving the info, the DMA Controller proceeds to perform the requested operation. When the entire block has been transferred, the Controller informs the processor by raising an interrupt signal.

↳ Fig: 4.18 shows an example of the DMA Controller registers that are accessed by the processor to initiate transfer operations



Two registers are used for storing the starting address and the word count.

↳ The third register contains status and control flags.

↳ The R/W bit determines the direction of the transfer.

↳ When this bit is set to 1 by a program instr:, the controller performs a read operation, that is, it transfers data from the mly to I/O device.

otherwise, it performs a write operation.

↳ When the controller has completed transferring a block of data and is ready to receive another command, it sets the Done flag to 1.

↳ Bit 30 is the interrupt-enable flag, IE. When this flag is set to 1, it causes the controller to raise an interrupt after it has completed transferring a block of data. Finally, the controller sets the IRQ bit to 1 when it has requested an interrupt.

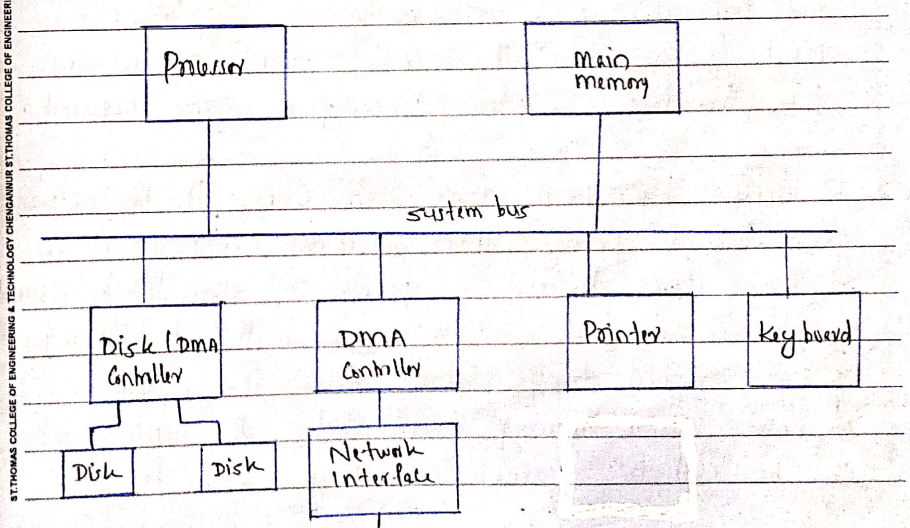


Fig: 4.19 Use of DMA Controllers in a computer system

An example of a computer system is given in fig: 4.19 showing how DMA controllers may be used.

To start a DMA transfer of a block of data from the main mly to one of the disks, a program writes the address and word count info into the registers of the corresponding channel of the disk controller. It also provides the disk controller with the info to identify the data for future retrieval. The DMA controller proceeds independently to implement the specified operation. When the DMA transfer is completed, this fact is recorded in the status and control registers of the DMA channel by setting the Done bit. At the same time, if the IE bit is set, the controller sends an interrupt request to the processor and sets IRQ bit. The status register can also be used to record other info, such as whether the transfer took place correctly or errors occurred.

Requests by DMA devices for using the bus are given higher priority than processor requests. Among different DMA devices, top priority is given to high-speed peripherals such as a disk, a high-speed network interface or a graphics display device. Since the processor originates most memory access cycles, the DMA controller can be said to "steal" memory cycles from the processor. Hence this interweaving technique is usually called cycle stealing. Alternatively, the DMA controller may be given exclusive access to the main memory to transfer a block of data without interruption. This is known as block or burst mode.

BUS ARBITRATION

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

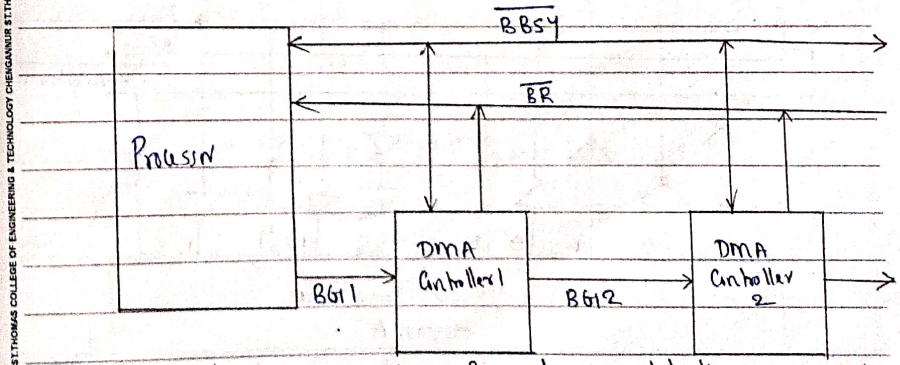
↳ Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.

↳ There are two approaches to bus arbitration

↳ In centralized arbitration, a single bus arbiter performs the required arbitration.

↳ In distributed arbitration, all devices participate in the selection of next bus master.

CENTRALIZED ARBITRATION



4.20 A simple arrangement for bus arbitration using a daisy chain

The bus arbiter may be the processor or a separate unit connected to the bus. In this case, the processor is normally the bus master unless it grants bus mastership to one of the DMA controllers.

A DMA controller indicates that it needs to become the bus master by activating the Bus-request line, $\overline{BR}$. When Bus-Request is activated, the processor activates the Bus-Grant signal, BGI, indicating to the DMA controllers that they may use the bus when it becomes free. This signal is connected to all DMA controllers using a daisy-chain arrangement. Thus, if DMA controller 1 is requesting the bus, it blocks the propagation of grant signal to other devices. The current bus master indicates to all devices that it is using the bus by activating the Bus-Busy line $\overline{BBSY}$.

DISTRIBUTED ARBITRATION

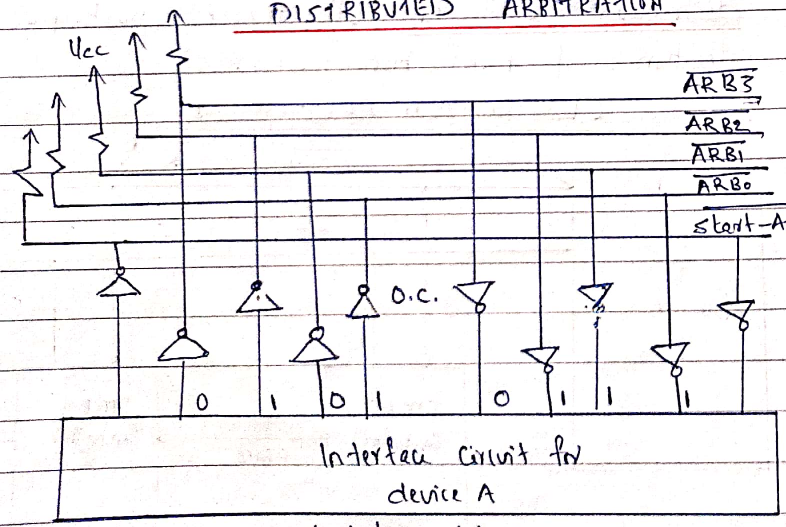


fig: 4.22 A distributed arbitration scheme

Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.

In fig:, each device on the bus is assigned a 4-bit identification number.

↳ When one or more devices request the bus, they assert the start-Arbitration signal and place their 4-bit ID numbers on four open-collector lines, $\overline{ARB0}$ through $\overline{ARB3}$.

↳ A winner is selected as a result of the interaction among the signals transmitted over these lines by all contenders.

↳ The net out-come is that the code on the four lines represents the request that has the highest ID number.

↳ The drivers are of the open-collector type. Hence, if the input to one driver is equal to one and the input to another driver connected to the same bus line is equal to 0 the bus will be in low voltage state. In other words, the connection performs an OR function in which logic 1 wins.

ST.THOMAS COLLEGE OF ENGINEERING & TECHNOLOGY CHENGANNUR ST.THOMAS COLLEGE OF ENGINEERING & TECHNOLOGY CHENGANNUR ST.THOMAS COLLEGE OF ENGINEERING & TECHNOLOGY

MEMORY SYSTEM

BASIC CONCEPTS

↳ The maximum size of the memory that can be used in any computer is determined by the addressing scheme.

↳ for example: a 16-bit computer that generates 16-bit addresses is capable of addressing upto 2^{16} = 64K memory locations.

↳ Data transfer blw the memory and the processor takes place through the use of two processor registers, usually called MAR (memory address register) and MDR (memory data register).

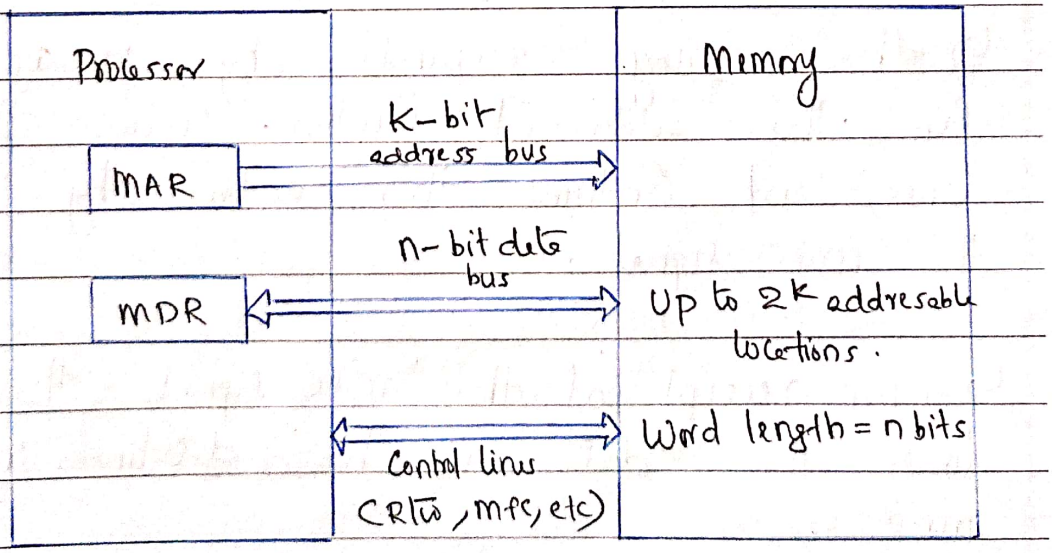


fig: 5.1 Connection of the memory to the processor

↳ If MAR is k bits long and MDR is n bits long, then the memory unit may contain upto 2^k addressable locations. During a mly cycle, n bits of data are transferred between the memory and the processor. This transfer takes place over the processor bus, which has k address lines and n data lines.

↳ The bus also uses the control lines R/$\overline{W}$ and mfc (memory function completed) for coordinating the data transfers.

↳ The processor reads data from the memory by loading the address of the required memory location into the MAR register and setting the R/$\overline{W}$ line to 1.

↳ The memory responds by placing the data from the addressed location onto the data lines, and confirms this action by asserting the mfc signal.

↳ Upon receipt of the mfc signal, the processor loads the data on the data lines into the MDR register.

↳ The processor writes data into a mly location by

loading the address of this location into MAR and loading the data into MDR. It indicates that a write operation is involved by setting the R/$\overline{W}$ line to 0.

MEMORY ACCESS TIME

It is the time that elapses between the initiation of an operation and the completion of that operation.

example: time b/w the Read and mfc signals.

MEMORY CYCLE TIME

It is the minimum time delay required between the initiation of two successive memory operations.

for eg: the time b/w two successive Read operations.

NOTE

The mly cycle time is usually slightly longer than the access time, depending on the implementation details of the memory unit.

SEMICONDUCTOR RAM MEMORIES

Semi Conductor memories are available in a wide range of speeds. Their cycle times range from 100 ns to less than 10 ns.

↳ Random - access memory (RAM) is a type of Computer data storage that is generally located on the motherboard.

↳ A RAM device makes it possible to access data in a random order, which makes it very fast to find a specific piece of information.

↳ RAM devices are used in Computer systems as the main memory (1° memory). RAM is a volatile memory, which means that the stored info: is lost when there is no power.

↳ So RAM is used by the CPU when a Computer is running to store info: that needs to be used very quickly but it doesn't store any info: permanently.

↳ RAM is the most common type of mly found in Computers and other devices such as printers.

↳ There are two types of RAM

1. STATIC RAM (SRAM)
2. DYNAMIC RAM (DRAM)

STATIC RAM

↳ memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories.

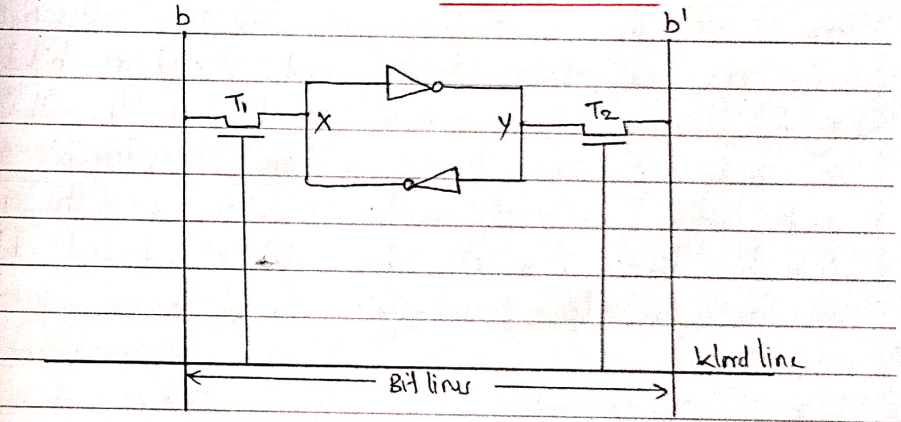


fig: 5.4 STATIC RAM CELL

↳ fig: 5.4 illustrates how a static RAM cell may be implemented.

↳ Two inverters are cross connected to form a latch. The latch is connected to two bit lines by transistors T1 and T2. These transistors acts switches that can be opened or closed under

Control of the word line.

↳ When the word line is at ground level, the transistors are turned off and the latch retains its state.

READ OPERATION

In order to read the state of the SRAM cell the word line is activated to close switches T_1 and T_2. If the cell is in state 1, the signal on bit line b is high and signal on bit line b' is low. The opposite is true if the cell is in state 0. Thus b and b' are complements of each other. Sense/write circuits at the end of the bit lines monitor the state of b and b' and set the o/p accordingly.

WRITE OPERATION

The state of the cell is set by placing the appropriate value on bit line b and its complement on b', and then activating the word line. This forces the cell into the corresponding state. The required signals on the bit lines are generated by the sense/write circuit.

CMOS cell (Complementary metal-oxide-semiconductor) Technology for constructing ICs (Integrated Circuits)

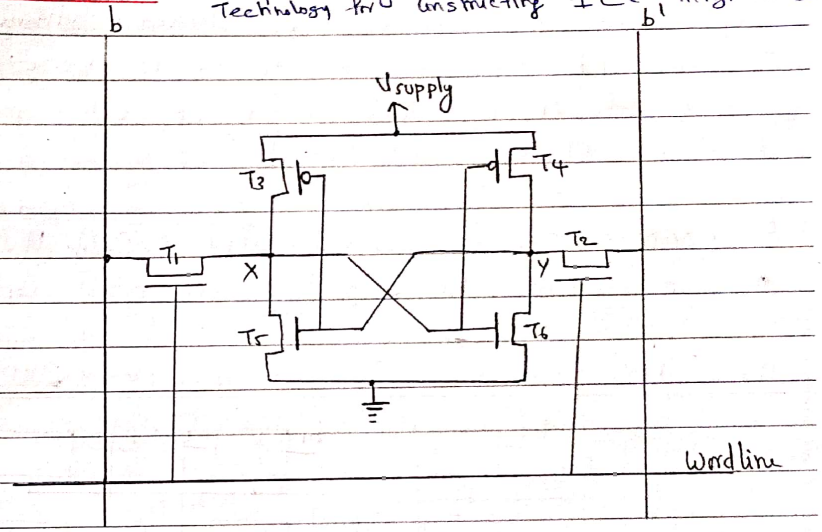


fig: 5.5 An example of CMOS memory cell

Transistor pairs (T_3, T_5) and (T_4, T_6) form the inverters in the latch. The state of the cell is read or written as explained above.

For eg: in state 1, the voltage at point x is maintained high by having transistors T_3 and T_6 on, while T_4 and T_5 are off. Thus if T_1 and T_2 are turned on (closed), bit lines b and b' will have high and low signals, respectively.

The power supply (V_{supply} voltage, V_{supply} is 5V in older CMOS RAMs or 3.3V in new low voltage versions.

A major advantage of CMOS RAMs is their very

low power consumption because current flows in the cell only when the cell is being accessed. otherwise, T_1, T_2, and one transistor in each inverter are turned off, ensuring that there is no active path between V supply and ground.

↳ SRAMs can be accessed very quickly. It is used in applns: where speed is of critical concern.

INTERNAL ORGANIZATION OF MEMORY CHIPS

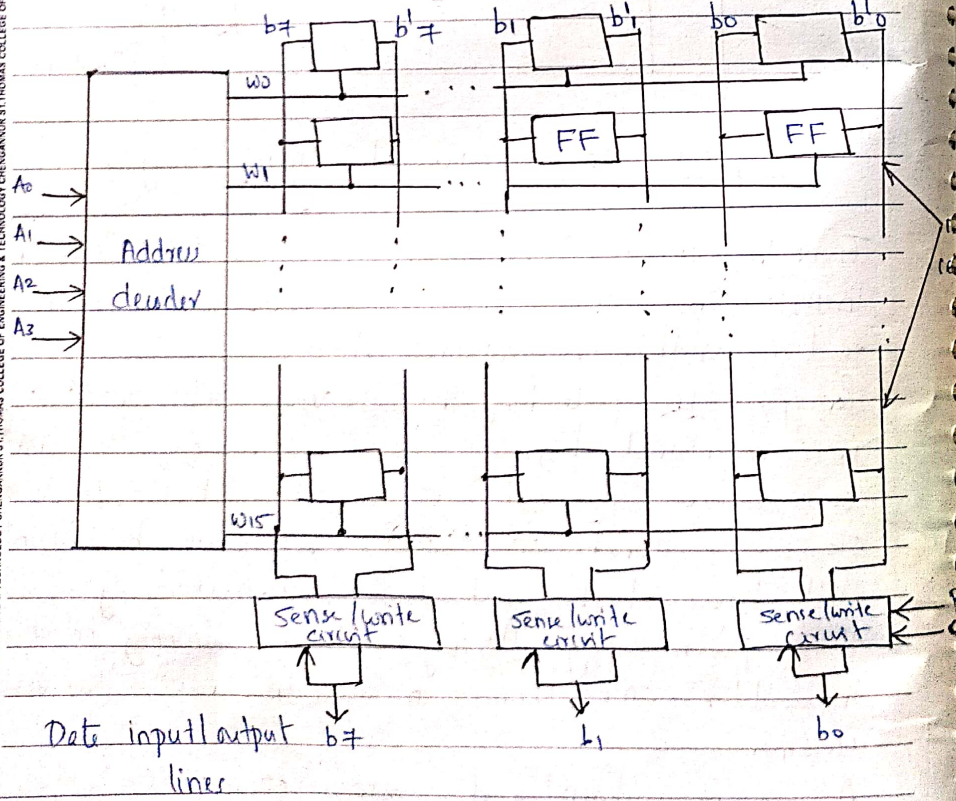


fig: 5.2 organization of bit cells in a mly chip

↳ memory cell are usually organized in the form of an array, in which each cell is capable of storing one bit of information.

↳ In fig: 5.2 each row of cells constitutes a memory word, and all cells of a row are connected to a common line referred to as the word line, which is driven by the address decoder on the chip.

↳ the cells in each column are connected to a sense/write circuit by two bit lines.

↳ the sense/write circuits are connected to the data input/output lines of the chip.

↳ During a read operation, these circuits sense or read, the info stored in the cells selected by a word line and transmit this info to the o/p data lines.

↳ During a write operation, the sense/write circuits receive input info and store it in cells of the selected word.

↳ two control lines, R/$\overline{W}$ and CS are provided in addition to the address and data lines. The R/$\overline{W}$ i/p specifies the required operation and the CS (chip select) i/p selects a given chip in a multichip mly system.

↳ fig: 5.2 is an example of a very small mly

chip Consisting of 16 words of 8 bits each. This is reffered to as a 16x8 organization. The data input and data o/p of each Sense/write Circuit are connected to a single bidirectional data line that can be connected to the data bus of a Computer. The memory Circuit stores 128 bits and requires 14 external Connections for for address data and Control lines. It also needs two lines for power supply and ground Connections.

ASYNCHRONOUS DRAMS

Information is stored in a dynamic memory cell in the form of a charge on a capacitor and this charge can be maintained for only tens of milli seconds.

↳ Since the cell is required to store info: for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value.

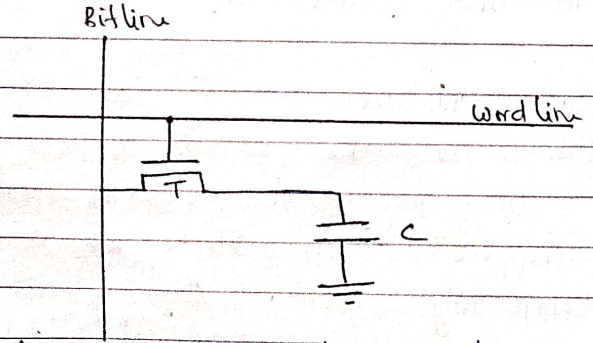


fig 5.6 A Single - transistor dynamic mly cell

↳ A example of a dynamic mly cell that consists of a capacitor C and a transistor T is shown in fig:

↳ In order to store info: in this cell, transistor T is turned on and an appropriate Voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor. After the transistor is turned off, the capacitor begins to discharge.

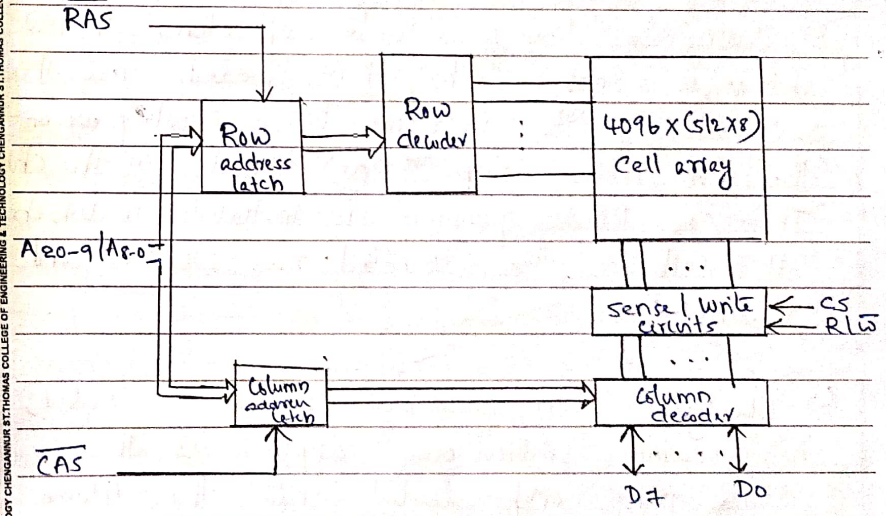


fig: 5.7 Internal organization of a 2M x 8 dynamic memory chip

↳ A 16-megabit DRAM chip, configured as 2M x 8 is shown in fig. The 4096 cells in

each row are divided in 512 groups of 8, so that a row can store 512 bytes of data. Therefore, 12 address bits are needed to select a row. Another 9 bits are needed to needed to specify a group of 8 bits in the selected row. Thus a 21-bit address is needed to access byte in this memory. The higher order 12 bits and the lower order 9 bits of the address constitute the row and column address of a byte respectively.

↳ During a Read or a Write operation, the row address is applied first. It is loaded into the row address latch in response to a signal pulse on the Row Address strobe (RAS) input of the chip. Then a Read operation is initiated, in which all cells on the selected row are read and refreshed.

↳ Shortly after the row address is loaded, the column address is applied to the address pins and loaded into the column address latch under control of the column Address strobe (CAS) signal. The info in this latch is decoded and the appropriate group of 8 sense/write circuits are selected.

If the R/$\overline{W}$ control signal indicates a Read operation, the o/p values of the selected circuits are transferred to the data lines D7-0

↳ for a write operation, the info on the D7-0 lines is transferred to the selected circuits.

In this DRAM the timing of the memory device is controlled asynchronously. A specialized mly controller circuit provides the necessary control signals RAS and CAS, that govern the timing. The process must take into account the delay in the response of the mly. such memories are called asynchronous DRAMs.

↳ DRAMs – low cost, high density.

FAST PAGE MODE

When the DRAM in fig: 5.7 is accessed, the contents of all 4096 cells in the selected row are sensed, but only 8 bits are placed on the data lines D7-0. This byte is selected by the column address bits A8-0. A simple modification can make it possible to access other bytes in the same row without having to reselect the row. A latch can be added at the o/p of the sense amplifiers in each column. →

↳ This scheme allows transferring a block of

data at a much faster rate than can be achieved for transfers involving random addresses. The block transfer capability is refferd to as fast page mode feature. (↓ The most useful arrangement is to transfer the bytes in sequential order, which is applied by applying a consecutive sequence of column address under the control of successive CAS signals)

SYNCHRONNS DRAMS

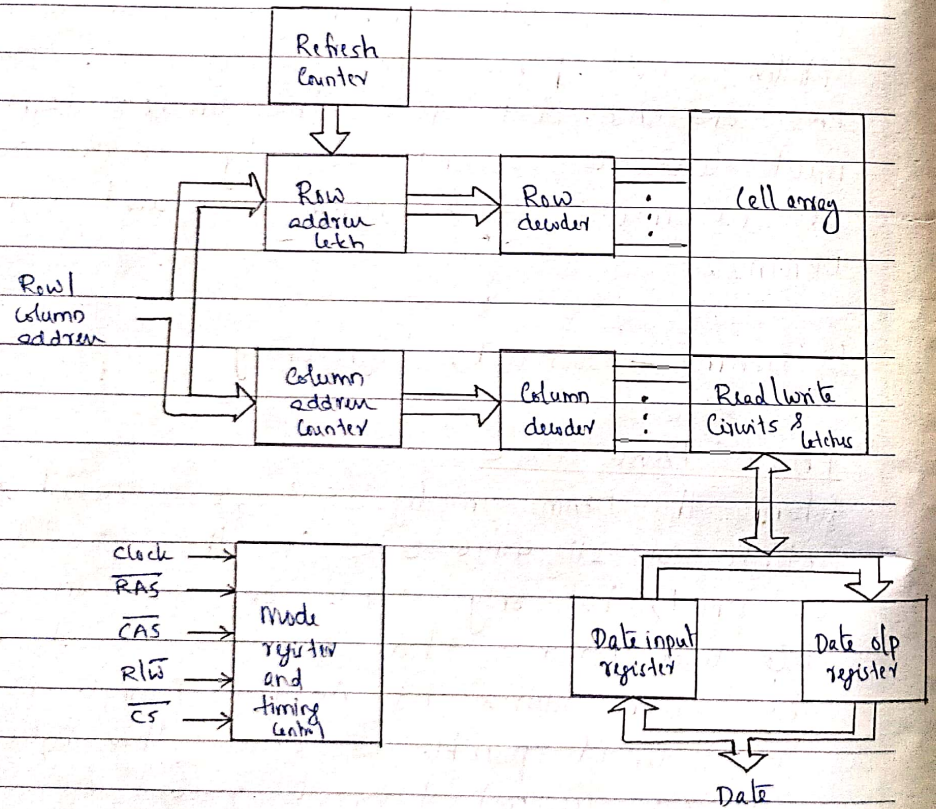


fig: 5.8 Synchronous DRAM

↳ more recent developments in memory technology have resulted in DRAMs whose operation is directly synchronized with a clock signal. Such memories are known as Synchronous DRAMs (SDRAMs)

↳ fig: 5.8 indicates the structure of an SDRAM

↳ The cell array is the same as in asynchronous DRAMs.

↳ The address and data connections are buffered by means of registers.

↳ A read operation causes the contents of all cells in the selected row to be loaded into the latches.

↳ Data held in the latches that correspond to the selected columns are transferred into the data o/p registers thus becoming available on the data o/p pins.

↳ The data from the data bus is transferd to read/write latches thrgh the data input registers. Also data from the read/write latches is transferd into the data bus thrgh a

date o/p register.

↳ SDRAMs have several different modes of operation, which can be selected by writing control info into a mode register.
eg: fast page mode, burst mode

LATENCY

The term memory latency is used to refer to the amount of time it takes to transfer a word of data to or from the memory.

BANDWIDTH

When transferring blocks of data, it is of interest to know how much time is needed to transfer an entire block. Since blocks can be variable in size, it is useful to define a performance measure in terms of the number of bits or bytes that can be transferred in one second. This measure is often referred to as the memory bandwidth.

MEMORY SYSTEM CONSIDERATIONS.

MEMORY CONTROLLER

↳ To reduce the no: of pins, the dynamic memory chips use multiplexed address inputs. The address is divided into two parts. The high-order address bits, which select a row in the cell array and the lower-order address bits which select a column in the cell array.

↳ A typical processor issues all bits of an address at the same time. The required multiplexing of address bits is usually performed by a memory controller circuit, which is interposed b/w the processor and the dynamic mly as shown in fig: 5.11.

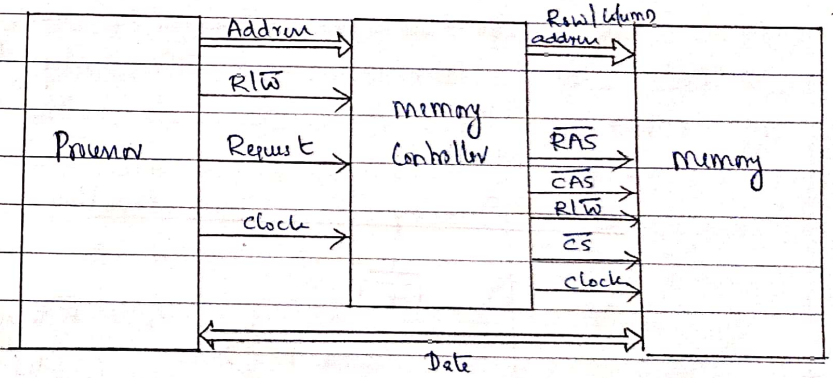


fig: 5.11 Use of a memory Controller

↳ The Controller accepts a complete address and the R/W signal from the processor, under control of a Request signal which indicates that a memory

access operation is needed. The Controller then forwards the row and column portions of the address to the memory and generates the $\overline{RAS}$ and $\overline{CAS}$ signals. Thus the Controller provides the RAS – CAS timing in addition to its address multiplexing fn:. It also sends the R/$\overline{W}$ and CS signals to the mfg. The CS signal is usually active low, hence it is shown as $\overline{CS}$ in fig:. Data lines are connected directly blw the processor and the mfg.

SEMICONDUCTOR ROMS

ROM (Read only memory) – stores info that can only be read.

↳ It is non-volatile storage.

↳ Data are written into a ROM when it is manufactured.

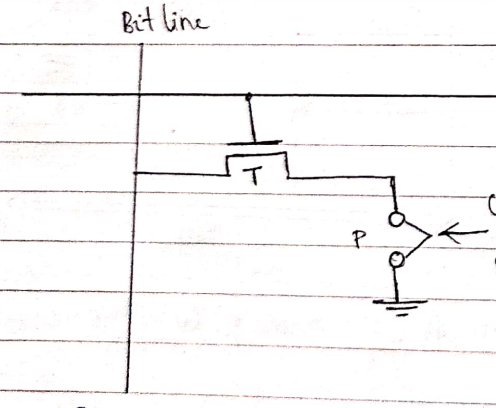


Fig 5.12 A ROM cell

↳ A logic value 0 is stored in the cell

if the transistor is connected to ground at point P, otherwise a 1 is stored. The bit line is connected through a resistor to the power supply. To read the state of the cell, the word line is activated. Thus, the transistor switch is closed and the voltage on the bit line drops to near zero if there is a connection between the transistor and ground. If there is no connection to ground the bit line remains at high voltage, indicating a 1. A sense circuit at the end of the bit line generates the proper o/p value.

PROM

↳ Some ROM designs allow the data to be loaded by the user providing a programmable ROM (PROM)

↳ Programmability is achieved by inserting a fuse at point P in fig: 5.12.

↳ Before it is programmed, the mfg contains all 0s. The user can insert 1s at the required locations by burning out the fuse at these locations using high current pulses.

↳ This process is irreversible.

PROMs provide faster and considerably less expensive approach because they can be programmed directly by the user.

EPROM

The ROM chip that allows the stored data to be erased and new data to be loaded. such erasable reprogrammable ROM is usually called an EPROM.

Erasure requires dissipating the charges trapped in the transistors of memory cells; this can be done by exposing the chip to ultraviolet light.

EEPROM

A significant disadvantage of EPROMs is that a chip must be physically removed from the circuit for reprogramming and that its entire contents are erased by the UV light. It is possible to implement another version of erasable PROMs that can be both programmed and erased electrically, such chips called EEPROMs, don't have to be removed for erasure.

- It is possible to erase the cell contents selectively.

The only disadvantage of EEPROMs is that different voltages are needed for erasing, writing and reading the stored data.

FLASH MEMORY

- A flash cell is based on a single transistor controlled by trapped charge, just like an EEPROM cell.
- In EEPROM it is possible to read and write the contents of a single cell.
- In a flash device it is possible to read the contents of a single cell, but it is only possible to write an entire block of cells.
- Flash devices have greater density, which leads to higher capacity and a lower cost per bit.
- They require a single power supply voltage and consume less power in their operation.
- The low power consumption of flash memory makes it attractive for use in portable equipment that is battery driven. Typical applications include hand-held computers, cell phones, digital cameras and MP3 music players.

MEMORY HIERARCHY

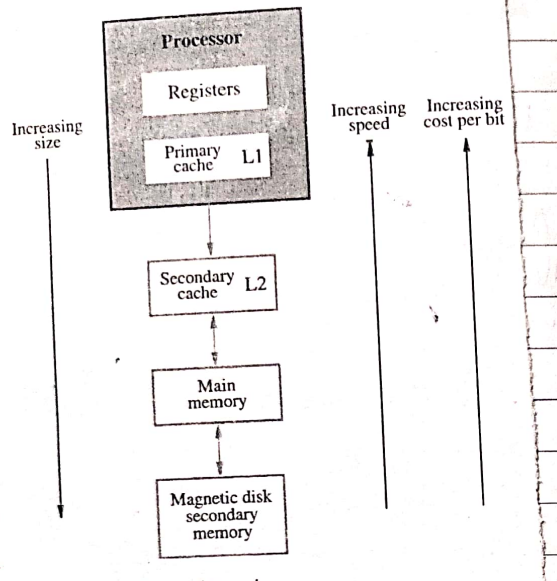


Figure 5.13 Memory hierarchy.

↳ The entire Computer memory can be viewed as the hierarchy. as shown in fig. 5.13

↳ The fastest accen is to date held in processor registers.

↳ The memory called processor cache holds copies of instructions and data stored in a much larger memory is provided externally.

↳ There are two levels of Cache

↳ A primary cache is always located on the processor chip.

↳ The primary cache is reffered to as level 1 (L1 cache)

↳ A larger, secondary cache is placed between the primary (ache and the rest of the memory.

↳ It is referred to as level 2 (L2) Cache

↳ It is usually implemented using SRAM chips.

↳ The next level in the hierarchy is called the main memory.

↳ It is much larger but significantly slower than the cache mly.

↳ In a typical Computer accen time for the main mly is about 10 times longer than the accen time for L1 cache.

↳ Disk drives provide a huge amount of inexpensive storage.

↳ They are slow Compared to the semiconductor devices used to implement the main memory.

CACHE MEMORIES

↳ The speed of the main memory is very low in comparison with the speed of modern processors.
↳ for good performance, the processor cannot spend much of its time waiting to access instructions and data in main memory.
↳ Hence it is important to devise a scheme that reduces the time needed to access the memory inf:.
↳ Since the speed of the main memory is limited by electronic and packaging constraints, the solution must be sought in a different architectural arrangement.
↳ An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster than it really is.

↳ The effectiveness of the cache mechanism is based on a property of computer programs called locality of reference.

↳ The point is that many instructions in localized areas of the program are executed repeatedly during some time period and remainder

of the program is accessed relatively infrequently. This is referred to as locality of reference.

↳ It is explained in two ways: temporal and spatial.

↳ The temporal aspect means that a recently executed inst: is likely to be executed again very soon.

↳ The spatial aspect means that instructions in close proximity to a recently executed inst: (with respect to the instructions addresses) are also likely to be executed soon.

↳ The temporal aspect of the locality of reference suggests that whenever an inf: item (inst: or data) is first needed, this item should be brought into the cache where it will hopefully remain until it is needed again.

↳ The spatial aspect suggests that instead of fetching just one item from the main mly to the cache, it is useful to fetch several items that reside at adjacent addresses as well.

↳ The term block is used to refer to a set of contiguous address locations of some size.

↳ Another term which is used to refer to a cache block – cache line.

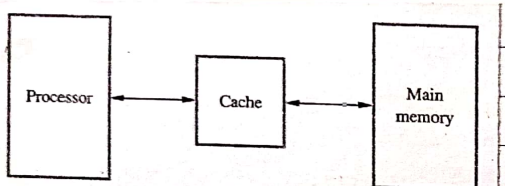


Figure 5.14 Use of a cache memory.

When a Read request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time. Subsequently when the program references any of the locations in this block, the desired contents are read directly from the cache.

The cache only can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory. The correspondence between the main mly blocks and those in the cache is specified by a mapping function.

↳ When the cache is full and a memory word (inst: or data) that is not in the cache is referenced, the cache control hlw must decide which block should be removed to create space for the new block that contains the referenced word. The collection of rules for making this decision constitutes the replacement algorithm.

MAPPING FUNCTIONS

To discuss possible methods for specifying where memory blocks are placed in the cache eg: Consider a cache consisting of 128 blocks of 16 words each, for a total of 2048 (2k) words and assume that the main mly is addressable by a 16-bit address. The main memory has 64 k words, which we will view as 4k blocks of 16 words each. We will also assume that consecutive addresses refer to consecutive words.

Direct Mapping

The simplest way to determine cache locations in which to store memory blocks is the direct mapping technique.

↳ In this technique, block j of the main memory

maps on to block j modulo 128 of the cache as shown in fig: 5.15

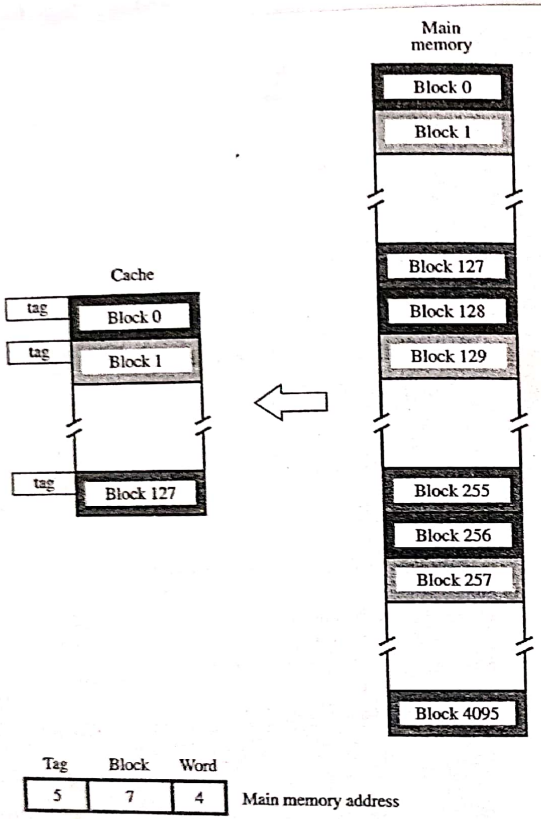


Figure 5.15 Direct-mapped cache.

Thus whenever one of the main memory blocks 0, 128, 256 ... is loaded in the cache, it is stored in cache block 0.

↳ Blocks 1, 129, 257, ... are stored in cache block 1 and so on.

↳ Placement of a block in the cache is determined from the mly address.

↳ the memory address can be divided into three fields.

↳ the lower order 4 bits select one of the 16 words in a block.

↳ when a new block enters the cache, the 7 bit cache block field determines the cache position in which this block must be stored.

↳ the high-order 5 bits of the mly address of the block are stored in 5 tag bits associated with its location in the cache. They identify which of the 32 blocks that are mapped into this cache position are currently resident in the cache.

↳ As execution proceeds, the 7-bit cache block field of each address generated by the processor points to a particular block location in the cache. the high-order 5 bits of the address are compared with the tag bits

associated with that cache location. If they match, then the desired word is in that block of the cache. If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache. The direct mapping technique is easy to implement, but it is not very flexible.

ASSOCIATIVE MAPPING

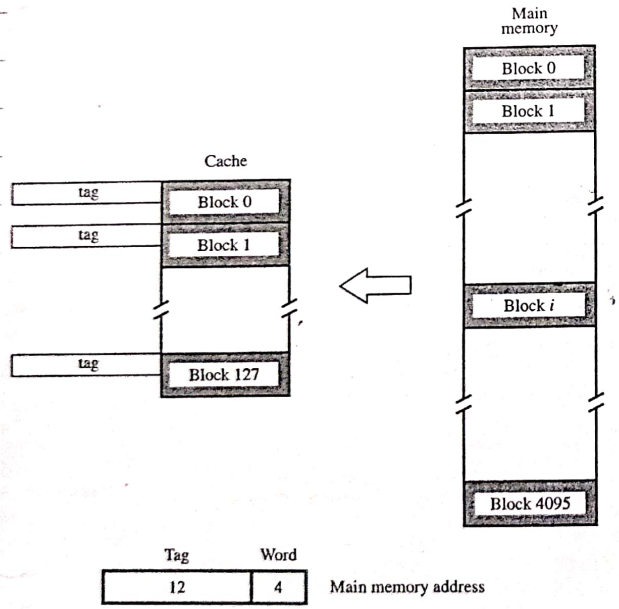


Figure 5.16 Associative-mapped cache.

In this case a main memory block can be placed into any cache block position.

↳ In this case, 12 tag bits are required

required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present. This is called the associative-mapping technique. It gives complete freedom in choosing the cache location in which to place the mly block. Thus the space in the cache can be used more efficiently.

↳ The cost of an associative cache is higher than the cost of a direct-mapped cache because of the need to search all 128 tag patterns to determine whether a given block is in the cache. A search of this kind is called an associative search.

Set-Associative Mapping

↳ A combination of the direct- and associative mapping techniques can be used. Blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set. Hence the contention problem of the direct method is eased by having a few choices for block placement. At the same time, the hdw cost is reduced by

decreasing the size of the associative search. An example of this set-associative-mapping technique is shown in fig: 5.17 for a cache with two blocks per set.

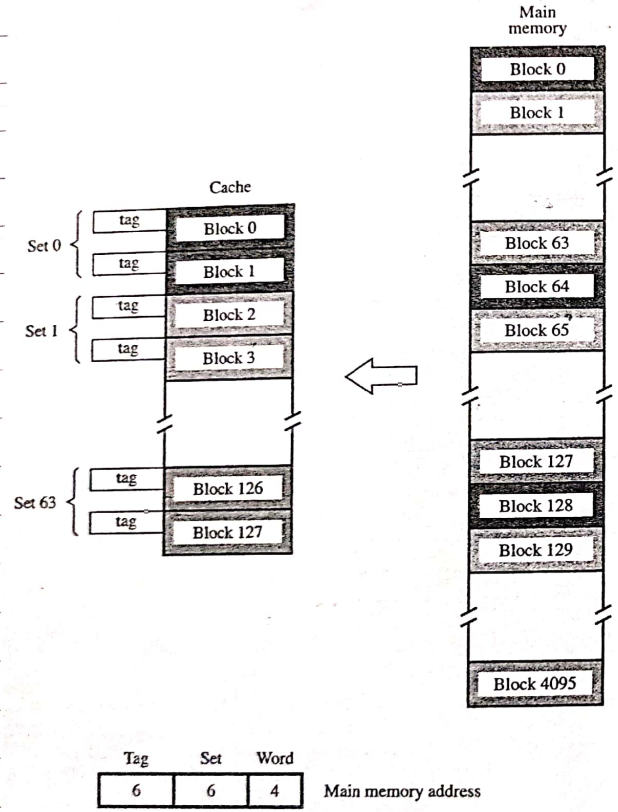


Figure 5.17 Set-associative-mapped cache with two blocks per set.

In this case, memory blocks 0, 64, 128, ... 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Having 64 sets means that the 6 bit set field of the address determines which set of

the cache might contain the desired block. The tag field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present.

Content Addressable memory (CAM)

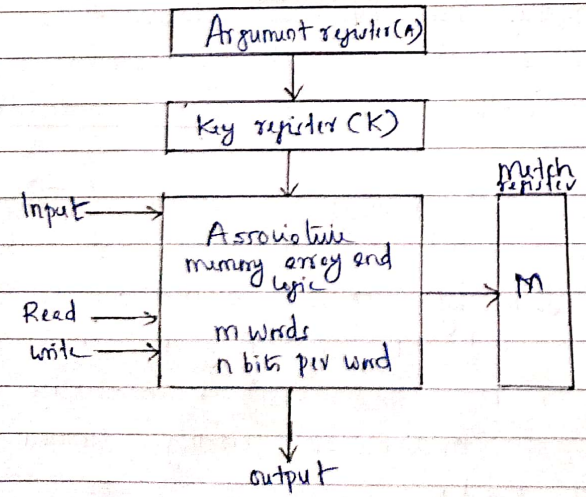
- Many data processing applications require the search of items in a table stored in memory.
- The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.
- A memory unit accessed by content is called an associative memory or content addressable memory. CAM.
- This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.
- When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word.
- When a word is to be read from an

associative memory, the content of the word, or part of the word, is specified. The memory locates all words which match the specified content and marks them for reading.

↳ An associative memory is more expensive than a RAM because each cell must have storage capability as well as logic circuits for matching its content with an external argument. For this reason, associative memories are used in applications where the search time is very critical and must be very short.

Hardware organization

Block diagram of associative memory



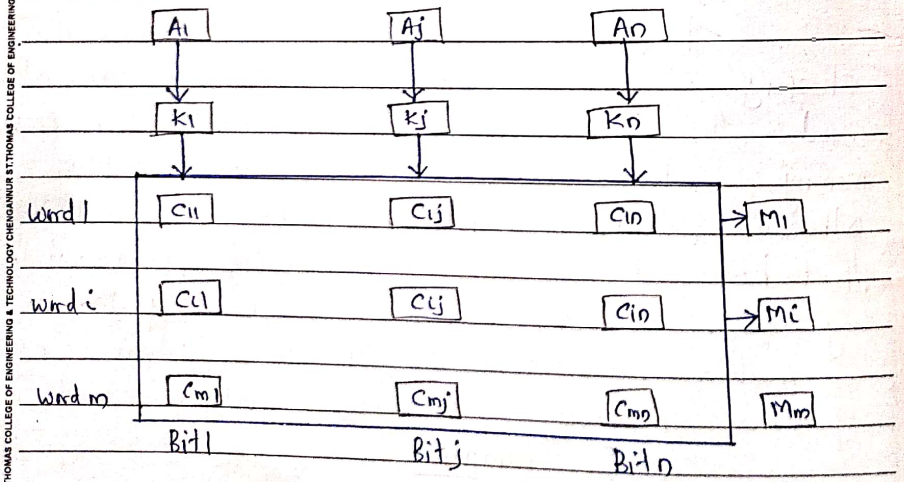
The block diagram of an associative memory is shown in fig. 12-6. It consists of a memory array and logic for m words with n bits per word. The argument register A and key register K each have n bits, one for each bit of a word. The match register M has m bits, one for each memory word. Each word in memory is compared in parallel with the content of the argument register. The words that match the bits of the argument register set a corresponding bit in the match register. After the matching phase, those bits in the match register that have been set indicate the fact that their corresponding words have been matched. Reading is accomplished by a sequential access to only for those words whose corresponding bits in the match register have been set.

The key register provides a mask for choosing a particular field or key in the argument word. The entire argument is compared with each memory word if the key register contains all 1's. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared. Thus the key provides a mask or identifying piece of info which specifies how the reference to memory is made.

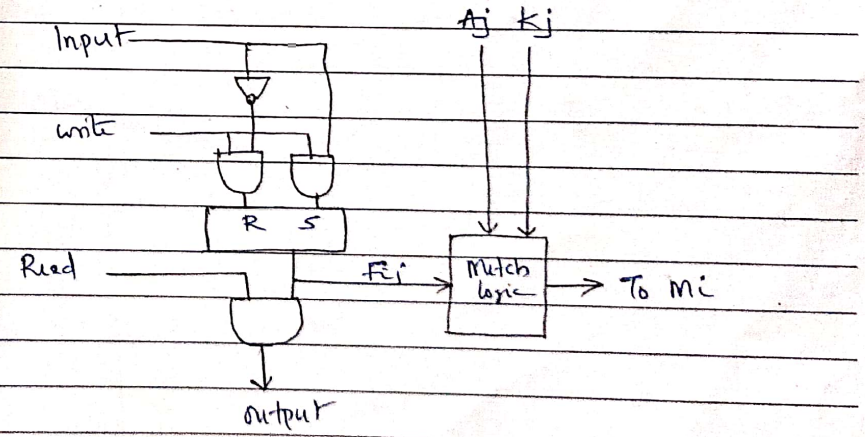
Example

A 101 111100
K 111 000000
Word 1 100 111100 no match
Word 2 101 000001 match

Only the three leftmost bits of A are compared with memory words because K has 1's in these positions.
Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.
The relation between the memory array and external register in an associative mly is shown in fig 12.7.
fig 12.7 Associative memory of m word, n cells per word



The cells in the array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. Thus cell Cij is the cell for bit j in word i. A bit Aj in the argument register is compared with all the bits in column j of the array provided that kj = 1. This is done for all columns j = 1, 2, ..., n. If a match occurs between all the unmasked bits of the argument and the bits in word i, the corresponding bit Mi in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.
The internal organization of a typical cell Cij is shown in fig. 12-8.



It consists of a flip-flop storage element Fij and circuits for reading, writing, and matching the

Cell. The input bit is transferred into the storage cell during a write operation. The bit stored is read out during a read operation. The match logic compares the content of the storage cell with the corresponding unmasked bit of the argument and provides an o/p for the decision logic that sets the bit in Mi.