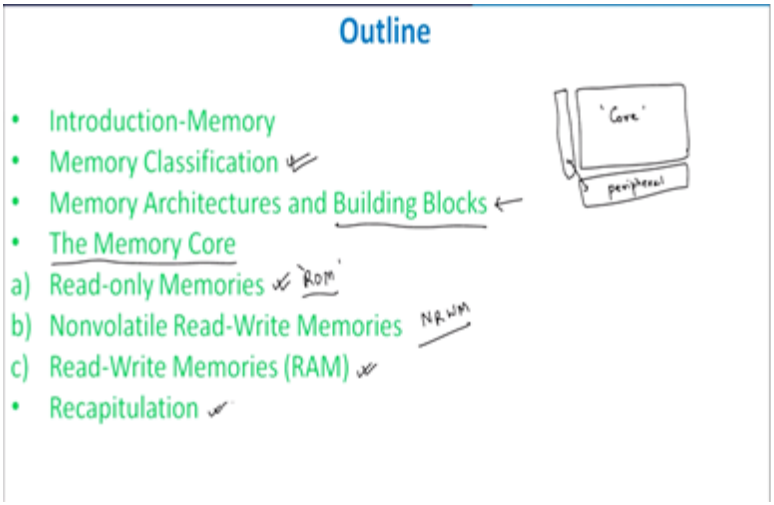
CMOS Digital VLSI Design
Professor Sudeb Dasgupta
Department of Electronics & Communication Engineering
Indian Institute of Technology Roorkee

Module No # 08
Lecture No # 39
Concept of Memory and its Designing - I

Hello everybody and welcome to the NPTEL online certification course in CMOS Digital VLSI Design. This course as we have already discussed till now we have finished part of synchronization which is part of this sequential circuits. We have also taken care of the various elements of sequential logic and combinational logic what we will be currently doing is one of the most important part of any digital circuit and that is basically a memory.

We in this module we will be trying to see how a memory functions what are the various types of memories and what are the various parameters of memory in using certain modifications on how we can improve the performance of the memory right. So these are the few things which you will be taking up in this module as far as this course is concerned. So the name of the module is conceptual memory and its designing. So we will be looking at designing aspects of memory as well, right. So the outline of the talk is something like this that we will be introducing you to the memory which is of course an important point we will look into memory classification

(Refer Slide Time: 01:31)



So the memory classification is this one. So we should know what are the various types of memories available to us and for what applications these memories can be utilized. Then we will look into a generic memory architecture so which means that if we have a memory what

are the generic architecture of a memory right. So that we can appreciate its functionality. In most of the cases memory is always taken as a block right.

And the memory block itself is plugged and played at various points across the network. So that is quite important that it is taken as a block but the understanding is that we will be going one step inside and see that apart from it is as a block how could I see it as a functional device right. And the various building blocks therefore were associated with the memory will be looking into picture.

As I discussed with you that at this part we will be looking into the memory core itself right. So typically when you go back to general design you have a memory code you will see the architecture later on and we will have peripherals attached to it right. So you will have, these are peripherals attached to it. What are these peripherals? These are peripherals right and these are peripherals, these 2 are peripherals for example and this is the core.

So memory core is the area where you store the data and the peripherals are those structures by which you access the data in both in terms of writing a data or reading a data right. So peripherals will be utilized for reading or writing the data whereas core will be primarily used for storing the data right. And therefore you will have multiple VDD lines as well to take care of it. We will also look into what is known as read-only memories which as ROMS, so we will look at ROM and what are the various configurations of ROM available to us.

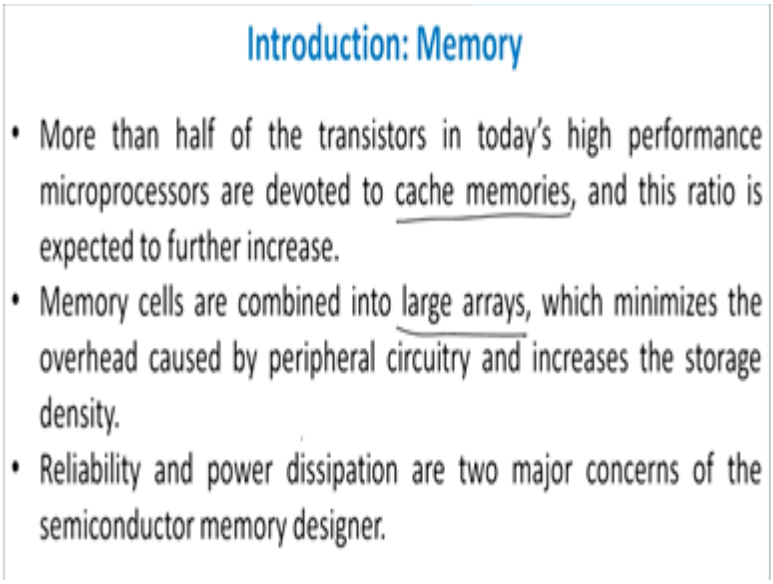
Then we will be looking into NRWMs which is basically Nonvolatile Read-Write Memories so these are nonvolatile read-write memories. We will go into RAM which is basically Random Access Memory as well as Read Write memory and then we will recapitulate the whole module in the generic sense.

So these are the few outline of the talk which we will be looking into. Now, what has happened is over the years if you actually look into the mood swap which tells me that for every one and a half years of growth in industry you will have approximately doubling the number of chips in a design which means that for every one and a half years of technology growth in semiconductor industry you should have at least.

Loading your dimensions of device by half. Then only you can double the number of active devices per unit area of the chip. This was holding good for moods for moods based devices so all your devices as well as your active devices all follows mood slot in larger extent. But

what happened to memory was, memory unlike your active devices memory per se does not follow mood slot as such right. It is slightly above the mood slot, what you will actually see is that the number of sells storing the bits will be much higher as much higher and over-estimated as compared to a mood slot.

(Refer Slide Time: 04:56)

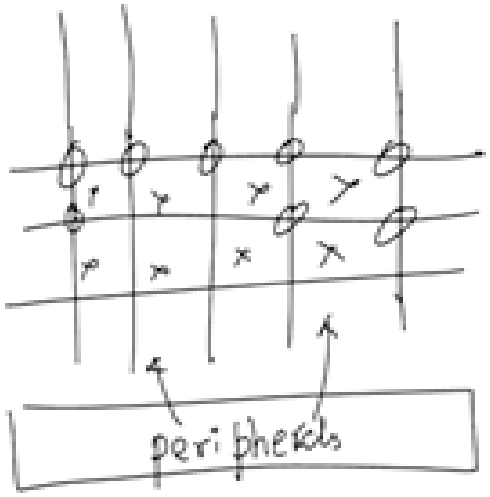


So what we are looking into is that in current microprocessor technology in current microprocessors or more than half of the transistors is basically cache memory right and that is quite interesting to look into it. And with passing times memory cache memory will increase in size and the processor itself will start decreasing in size right. So we are looking into a structure where memory will be quite a large space will occupy a larger space and therefore its optimizations will be quite important.

So memory cells are therefore combined into large arrays as I told you and therefore the so what is does so you see as I was discussing in the previous slide that if you have a core and a peripheral right but if you had one memory element here and another memory element so on and so forth and for each memory element you had a peripheral here so you require large amount of peripherals so peripheral swill take large amount of area right.

So what people thought was like when they were actually organizing the the structure they thought that if we can therefore have the code form of an array then for an single array cell I can choose a single peripheral right. And therefore the number of peripherals required will be quite minimized so that was the prime motivation why your memory cells were arranged in an array right.

(Refer Slide Time: 06:26)



Array means basically they are in the form of like this you will have sorry so I will just show it to you so array basically means you will have rows and columns so this one is column and this one will be row right so what this is your memory array so this will be your peripherals here, peripheral right, pherals. So they will be driving these memory course right so this is the memory core for here is the memory cores here which is available and they are able to store a data and or what or the core which you see.

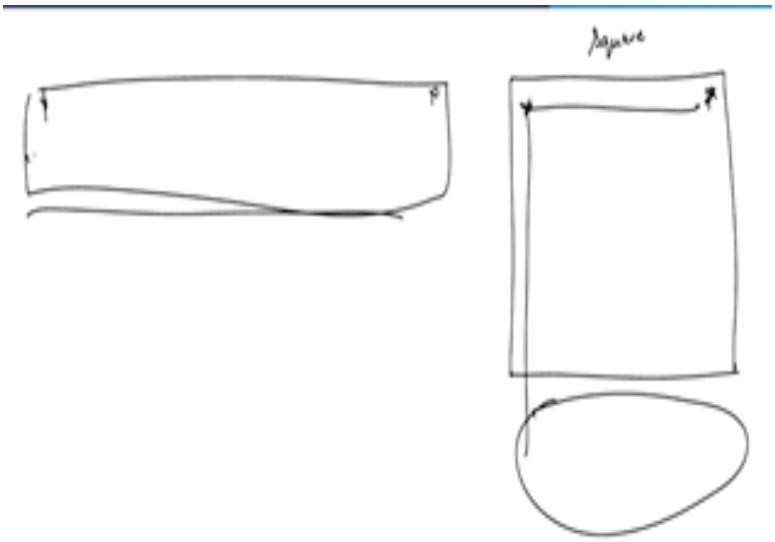
And you can store a data or you can retrieve a data, and this peripherals you can actually retrieve the data in emergencies. So you see once you make it in a form of an array the number of peripherals required drastically reduces right and that was the reason why you can actually have a larger storage diversity which means that you concentrate more on the storage area rather than on the peripheral networks as such.

And that is quite an interesting area of memory design. Two important concerns memory is because it is an arrayed architecture so if you do a mistake in one of the cells that mistake might be repeated in relatively all the cells and as a result reliability is a major issue in a memory design apart from that since memory will be accessed by an external source they will be large switching activity.

Which will be available to you. And therefore its power decapitations will also so very high. So, from 2 major concerns of memory design is its reliability and its power gun, power dissipation or power consumption whatever you want to say. These 2 will be the 2 major factors which determine the memory elements of the memory issues right so what we do is

in typically when you try to design it that you first design a smaller array and then you just replicate the smaller array into larger array

(Refer Slide Time: 08:28)



So the most typical way of looking at it is that generally we required that we should have a square profile or the rectangular profile of the memory. The reason being that is the idea was that you should keep the memory core in a square shape generally so that you can access any data like the access times for any data is almost equal to each other. So if you have a data placed here and a peripheral which is kept here then the amount of time taken to access this data

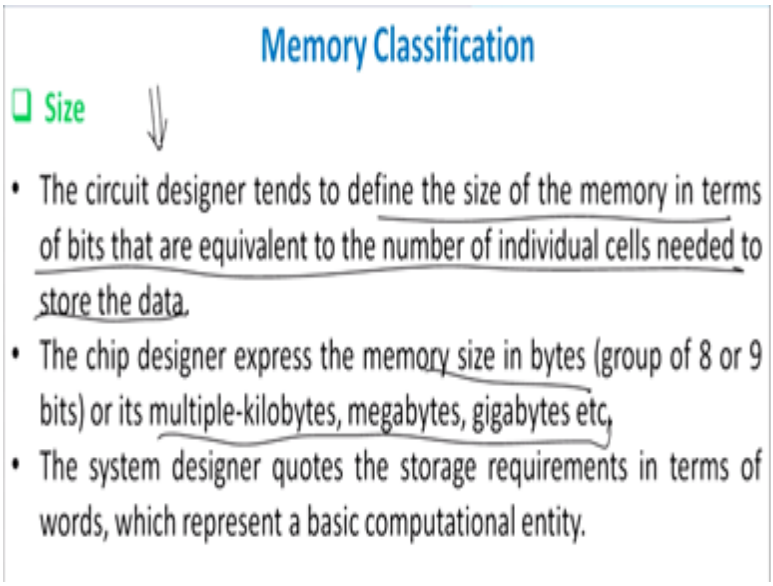
And this data will be approximately equal to each other whereas if you have a rectangular sort of a this thing then this data and this data it will take large amount of difference in time right so those things are very important as far as its memory is concerned. So memory core is typically a square we generally like to keep it square in dimension reliability and participation is the major issue because each cell will be accessed the third very important issue.

Is that when you store a data in a memory that is generally stored in form of a charge right so you have to ensure that the charge does not leak away when it is storing a data so that is also very important part. So whenever we were looking into static RAMs for example RAM and DRAM we will see that in certain cases you require to refresh the memory so the charges get refreshed.

And this even if there is a leakage it gets refreshed and you are able to store a data where in some cases you when you actually do not refresh it but you apply certain techniques by which

you do not let the charge shift away from an active note. So these are the few important points which should one should be aware of and one should be aware of its this thing.

(Refer Slide Time: 10:05)



So what so if you look at the first part here when we are concentrating on the size the circuit designer tends to define the size of the memory in terms of bits that are equivalent to the number of individual cells needed to store the data. so it means that if you have ten cells it can store 10 bits of information right, ten zeros or ten ones or combinational ones and zeros right.

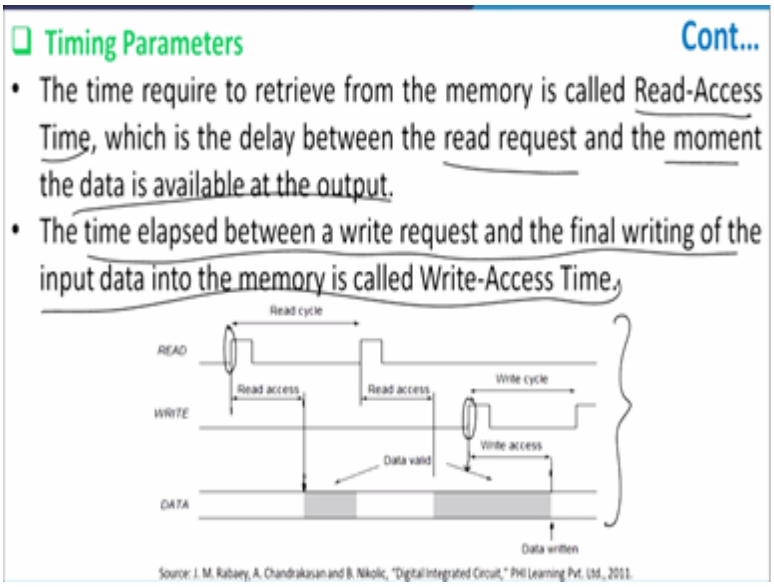
So, when you have say tens right you can store ten bits if you have therefore 10 KB design 10 KB memory is a actually 10 into 10 to the power of 3bits of data which 10 to the power 4 data bits of data can be stored in a design right so that is the reason you generally therefore the size of the memory is exactly equal to the number of bits how a memory can store the chip designer expresses the memory size in bytes.

We already know this point on its multiple kilobytes, megabytes, gigabytes that we have already seen right. From a system designers point of view we will be recalling the word rather than byte itself but from a devise engineer or from a engineer who is working at a memory level we will be accessing the bytes from the peripherals whereas from system designer.

Maybe the word is basically the system designer will code the storage the requirement in terms of words right which depicts the basic computation entity right, so if you ask somebody that I want to store a word of say I want the memory to be of ten megabyte so you need to

require to have ten to the power 6 number of cells to store that pin number of bytes properly right.

(Refer Slide Time: 11:42)



So this is quite for memory it is quite important and issue which is important now let us look at some of the parameters of the memory right which means that what are the various parameters of the memory which is available to us. These parameters are first is the read access time right so this is the difference or the time taken between the read request and the moment the data is available at the output so.

Let me show you from this diagram that which is a timing diagram that you see in it at this point so what it does is that the peripheral will reset the read request right and there will be some difference between the read request and the moment the data is available at the output. That difference is basically known as the excess time. So if you look very carefully here the read request is available to at this point of time right at this point.

When the read voltage goes high you are sending a request to read a data from the memory core. And actually the data is read somewhere in here right. It is coming to the output somewhere this point so time in time domain the difference between this point and this point is defined as my write access time right. So read access time is this is what we get it so whenever you have read high to a valid data available to you in the output side is basically defined as a read access time right.

And the time elapsed between the a write request and the final writing of the input data into the memory is called the write-access time. So I will just show you, when you get write access here so you see if you look very carefully this is basically my write access so right signal is gone high and the data is actually written at this point right**.** And the difference between this and this is basically defined as my write access. So I will have read cycles, I will have write cycles right and the read cycle and write cycle will be complementary in nature.

And therefore I will be at one point of time reading bits from the input and in other case I will be writing bytes or writing information onto the system. The difference between the read request and actual data in the output available in the output is my read access time and what is write access time when the write request is actually being sent by the peripheral and the time at which it actually writes the byte onto the memory cell is defined as my write access time.

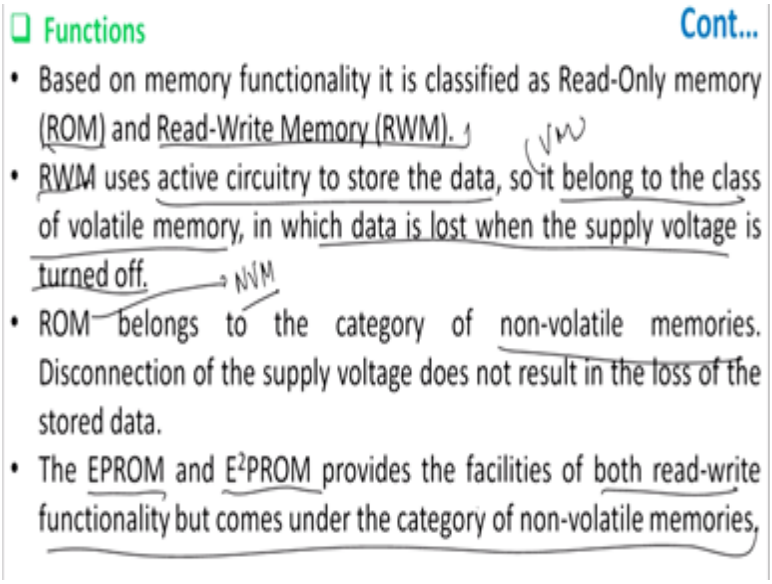
So this is what that is what read access time and write access times are all about. As I discussed with you read and write typically generally or generally perfects systems should not be over writing towards each other. So you will have read and write cycles which are quite complimentary in nature right and they will be quite different in terms of write one has to be very careful here about 2 things that.

This delay between the read request and the actual reading of the data is primarily because of the intrinsic delay of the gates available in the system plus inter collect delay which was there between the systems right. So that is the reason you will always have a rig margin. Second thing is that please understand while reading a data you should be very careful that you don't destroy the data itself right.

So you are reading a 1 from a cell but then you do not destroy the one from the cell itself right, while writing a data similarly I should ensure that I have written the data properly not only that for example I want to write one there then my VVD should be so high and the should be actually written in the data right. And these things one should be quite careful about as far as dealing with the system is concern.

Now as I discussed with you in the first slide itself that you will have based on the memory functionality right it can be classified as ROM, read only memory and read write memory.

(Refer Slide Time: 15:42)



So you will have Read only memory here which is basically my ROM right and you will have RWM which is basically my read write memory. So you will have a read write memory which is basically writing, reading and writing the cell and you have only reading. So in a ROM you generally do not write anything to it, you have already written initially at one point of time and then you only access to it.

For example your OS which used to store right you start up OS which uses to stored in the system they are all ROM right. And therefore you cannot change the ROM and therefore once it has been written it is written and you can only access the data from the external world. RQM which is basically the read write memory uses active circuitry to store the data.

So it belongs to the class of volatile memory in which data is lost when the supply voltage is turned off. So since in read write memory or um basically a RAM or for that matter any of the systems, when you store a data and it is part of a volatile memory so there are 2 types of memory volatile and non-volatile. Volatile memory is one memory when you switch off the power the memory actually loses.

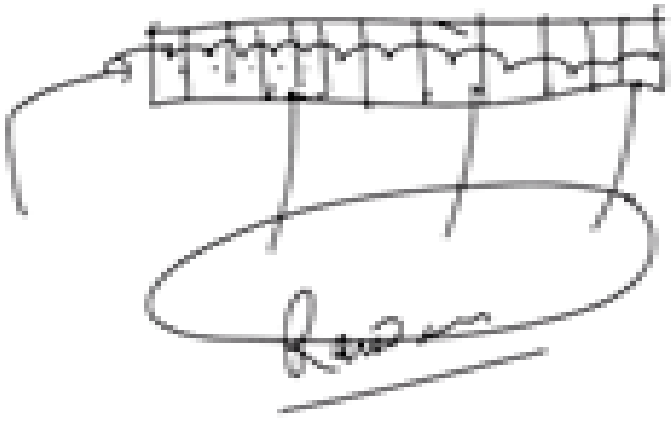
Its data and non-volatile memory are those which you whenever you switch off the power you do not lose the data as such. This RWMs are part of volatile memory whereas ROM belongs to a part of non-volatile memory so they are basically NBMs. So this ROM is actually NVM, Non-volatile memory whereas this RWM is part of VM volatile memory right and therefore disconnection supply voltage result in the loss of the stored data.

That is quite interesting and important that in this case in ROM case you do not lose data whereas in RWM you start losing data when you switch off the power right and that is quite critical in the understanding of the power systems There are 2 issues which we combine erasable programmable read only memory EPROM and EPROM provides the facilities of both read-write functionality but comes under the category of non-volatile memories so E2PROM and EPROMs are basically erasable programmable read only memory.

Executable erasable programmable read only memory and both comes under NVM and therefore when you switch off the power you actually lose data in this case right. And you are able to lose the data very fast in this case but both provide read write facilities functionality. So depending upon how you access your data we can therefore segregate the memories most of the memories.

Which you will study through your books or even this tutorial or anywhere else is primarily random access memory so these are actually RAMs right. Which means that you access, what is the random access memory means you do not sequentially access the data, I can access any data, any point within the memory core that is the what is known as random access memory. So that is what is written here is in which the memory location can be read and written in a random order. That means I can read or write in a sequence.

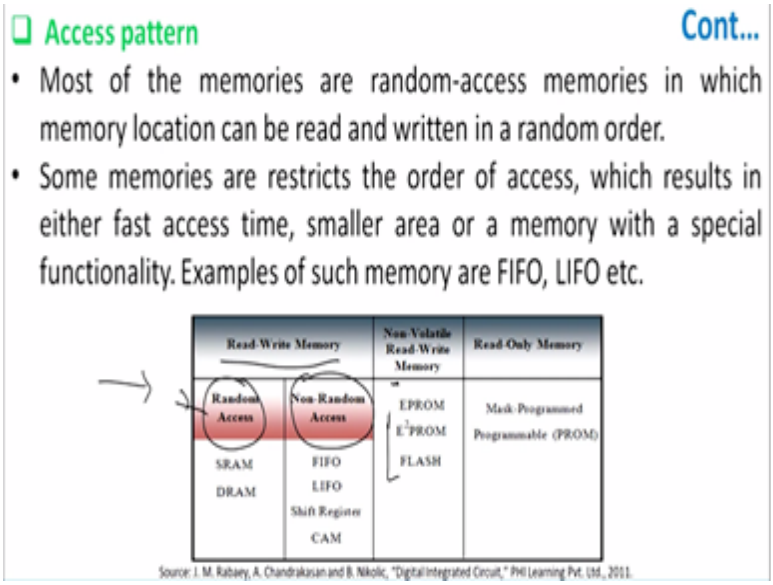
(Refer Slide Time: 19:01)



So I will just give you an example say we are doing a sequential memory so I have got a cell here and how I do that is just like a shift register, I go on writing here right I have got say 1, 2, 3, 4, 5, 6, 7, 8, 9, ten, eleven, twelve right. So what I do is with clock I suppose then this

data is written here sequentially then it goes to this point then it comes to this point this and it makes a hop sequentially and reaches this point right. Whereas you can do random writing and by feeding the information directly to this so this is known as basically random access and this is my sequential access right and that is one should be quite aware of.

(Refer Slide Time: 19:43)



Some memory are restricted have restricted orders of access which results in other fast access times, smaller area or a memory with a special functionality. I will give an example say you want to reduce your access times right then what I try to do is that rather than accessing all the cells I might only be able to access few of the cells. When I once I access few of the cells. Then I pretty well reduce the read access time as well as the write access time right and the price you pay for it is not all the memory cells are actually being evoked at any one point of time.

So examples are FEFO and LEFO which is first in first order, last in first order these are basically your memory with certain functionality. So now if you look at this table which is in front of you we have a RWM which is read write memory which in which we have random access memory, non-random access memory. FIFO and LIFO are part of an non-random access memory.

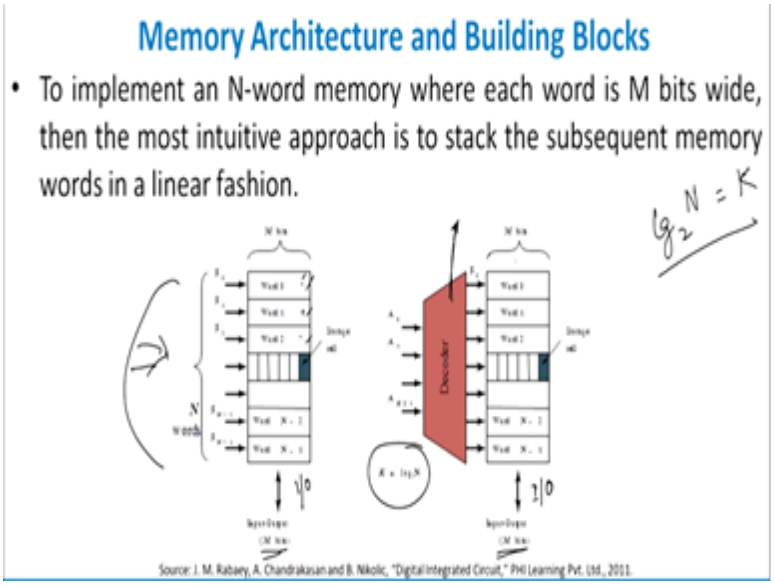
FIFO and LIFO are part of an non-random, FIFO means first in first out so basically its sequential writing of data into the memory and LIFO is or shift register or CAM is basically sequential writing of memory in the system. And random access is when you have SRAMs and DRAMs available right in SRAMs for example static random access memory or dynamic

random access memory in both the cases you get access any memory cell at any point of time right.

And therefore you will always have a random access here and you will have a non-random access here in this case. Another is basically non-volatile read write memory and there are 3 of them, EPROM, E2PROM and FLASH, so there are the 3 memories that are available again non-volatile so whenever you switch off the data you do not actually change the switch off the power supply you do not change the content.

Within the cell and read only memory basically ROM and these are basically programmable and mass programmable designs. So, PPROMS programmable read only memory and what are these programs you program it once and it stores the data and so every time you want to run the profile, run the PROM it does in a proper sequence so that your program is loaded in a proper manner right and that is what people have been doing in this case.

(Refer Slide Time: 21:57)



Now let us come to the memory architecture right and the building block for the memory architecture so let us suppose we want the implement N-word memory where each word is M byte wide so I have a N word memory 1 so if you look it very carefully it is word zero, 1, 2, so on and so forth and each one of them will have M bytes right. So, let us suppose 8 byte memory there are such N type of 8byte memory available then the best fashion to do it is slack them, slack the words like this.

So, you slack them like this right and you access so there are N words here if you look very carefully these are the N words which you see these are the N words and this is basically

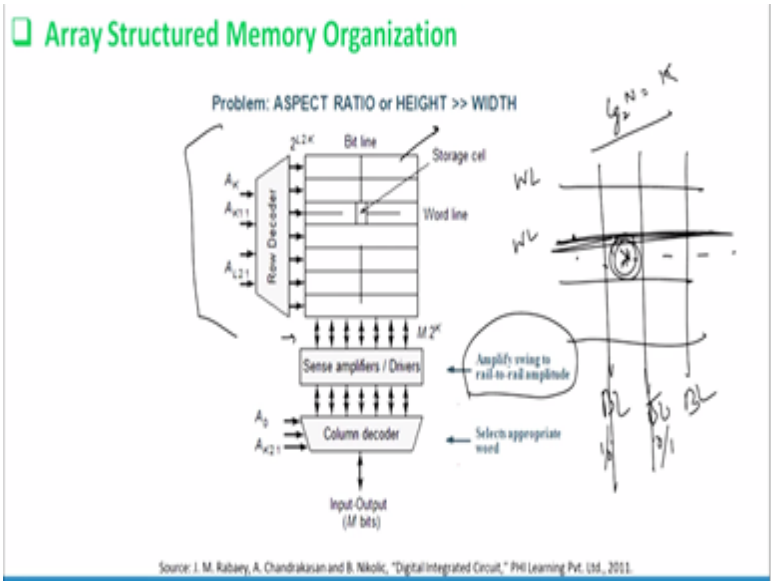
MIO ten bytes right with N bytes. Now what people did was that they use a decoder here if you use the second diagram here the user decoder why this decoder was used? The decoder was used, you want to select a particular word right to access a bit say I want to access fifth byte of third word, let us suppose then you have to access the third word line and then you have to access the fifth byte out of it.

So what you people have done is that they have used a what is known as a decoder unit here, what does decoder unit do is that it reduces the number of inputs available to you to catch hold of the particular output byte. Which means that, let us suppose that the word was hundred then you do not actually require hundred number of IOS to write a cell. You require much smaller than that basically and the relationship is given as K equals.

To log N is typically the relationship which you see log N to the base 2. So log N to the base 2 is basically like K. so if you have any equals to say a hundred then log hundred to the base 2 will be your K value. So that means the number of cells number of inputs primary inputs can use it right. And that reduces drastically the expected inputs on the system this makes it stored cell much easier to handle and you will have therefore input bytes.

Input output you will also and IO byte here there will have consisting of N bytes here and this will also have N bytes here and there will be decoder unit which is basically this much and there are N bytes here so there are N words, M bytes and there will be a decoder which will take care of the, so this is basically your also known as your row decoder in the case. Because it is actually decoding the row, the various rows available tries to select the particular row right where you want to access or write the data right so this is the primary architecture most institutively people have been doing it for a long duration of time.

(Refer Slide Time: 24:44)



So if you look very carefully, if you look back this was what we have already done right, this part we have already seen, that you have a memory core, this is the memory core and this is the storage cell you have row decoder here right and you will have column decoders here. I will explain these terms to you, these are row decoders which are primarily responsible for decoding the rows available here and it is basically log end.

To the base 2 as I discussed with you the number of inputs to this cycle so you will have much smaller inputs here to access the storage cell. So let us look at what happens in the column part. You actually require example, what is the amplifier? Well it is pretty simple and straight forward but if you have a 1 let us suppose you are storing 1 on to a cell right, you have 1 storing onto a cell and you want to access it, so once you access it the voltage and the output start to rise near vividly right. Since amplifies helps to see the difference between that voltage and the base voltage and amplifies it so the rise times are very fast and therefore your access times are reduced drastically right.

So that is the reason you require sense amplified and it also helps you to give a rail to rail amplitude to a larger extent, amplify swing to rail to rail so you will have full zero equity if you are using wheel zero as the ground and dimity as your output you will accessing zero dimity full rail to rail swing using the sense amplifiers right. And they will also reduce your accessory drastically.

Then you have a column decoder, same concept again the column decoder will look into the column and the relationship also happens to be the log end to the base equals to K and from there you get the column decoder available. So column decoder are attached to sense

amplifiers as sense amplifiers drives circuit so, if you look very carefully you will have a architecture something is like this so you will have word lines.

Here so these are word lines and you will have byte line and byte line so this is byte line and this is suppose byte line bar, similarly this is byte line here right. So if you want to access the cell here let us suppose and this byte line and this byte line bar will one zero or zero one will be stored here. so what you need to do is you need to make this word line high right so you are able to access all the cells here and these byte and byte.

Bar line will access the cell which is the common part between the byte line and the byte line bar fine. So this is what generally the general architecture of array structure of memory happens to be in a detailed manner. Generally the height should be slightly larger than the width so this height which you see in front of you will be slightly larger than the width right. And thus is the standard width of looking at it.

Then similarly if there are M bytes of information which you want to find out then M into 2 to the power K will be the total number of sense amplifiers that drivers require to actually access the byte between the memory cell right. While K is the basically that this value this is what is K right. So you very well know how many number of words are there and from there you can find out the K. This K can be fed to find the value of the total number of sense amplifiers required right. With this let me stop here today and then we can carry on with the other ceremony later on thank you.