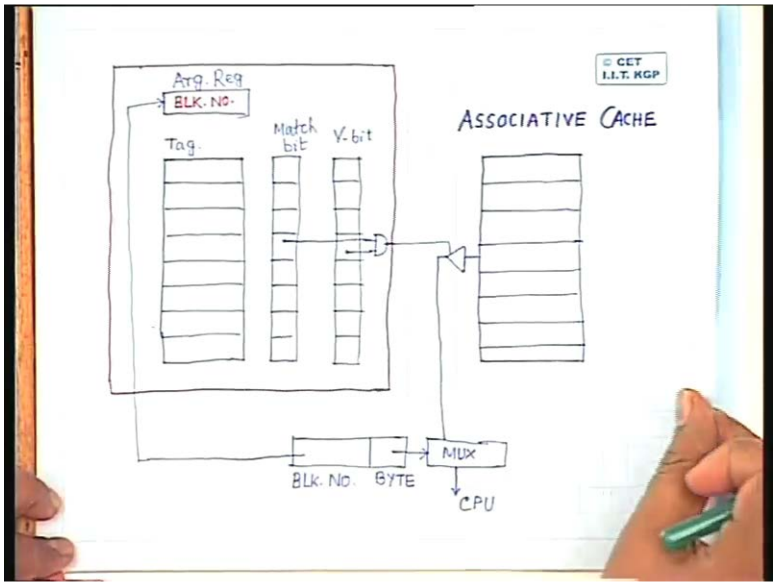
Digital Computer Organization

Prof. P.K. Biswas
Department of electronic & Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Lecture No. # 19
Cache Memory Architecture RAM Architecture

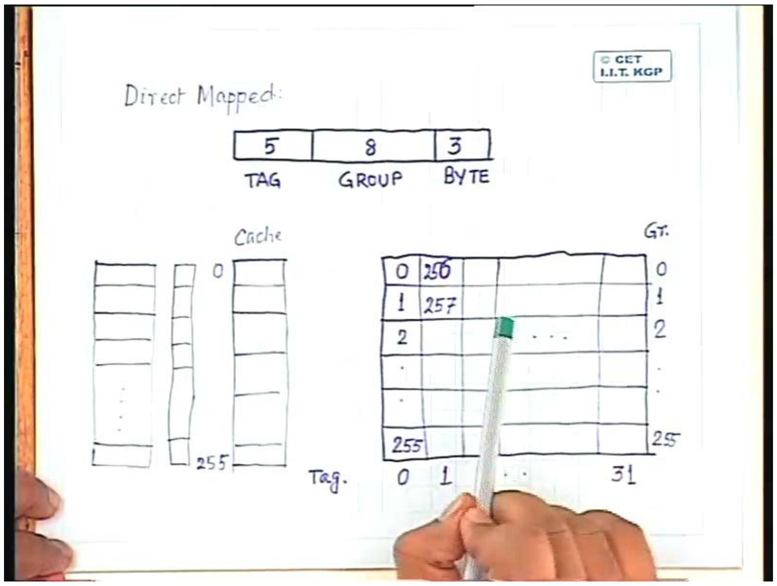
So, we started our discussion on the cache memory. In the last class we have discussed about the associative cache and the direct mapped cache.

(Refer Slide Time: 00:01:50 min)



In case of associative cache, we have seen that a block from the main memory can be placed in any of the cache lines. The block address will be placed in the corresponding tag memory and the valid bit in this tag assembly has to be made equal to 1. So whenever we want to search for a particular block whether that block is available in the cache memory or not, the block number has to be compared to each of this tag memory locations in parallel and depending upon the output of this match result and whether the corresponding valid bit is equal to 1, you take out the corresponding cache line and select one byte from the cache line through a multiplexer, by making use of the byte identification field.

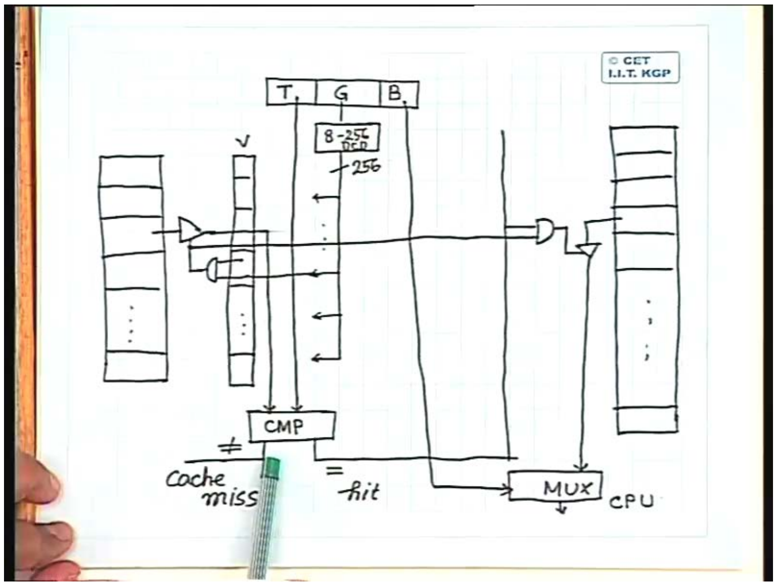
(Refer Slide Time: 00:02:06 min)



Whereas in case of direct mapped cache we have said that a particular block of the main memory can go into a particular cache line; unlike in case of associative cache where a main memory block can go in any of the cache lines. In case of direct mapped cache a block in main memory can go only to a particular block in the cache, it cannot go anywhere else. The advantage that we have got in case of direct mapped cache is the reduction in hardware, whereas in case of associative cache if there are n number of cache blocks or cache lines, I have to have n number of comparators. Whereas in case for direct mapped cache, as we have seen with the help of the schematic diagram that the n number of comparators is replaced by a single comparator. And this is possible because of a limitation that one block of main memory can go only to one cache line, it cannot go to any other cache line.

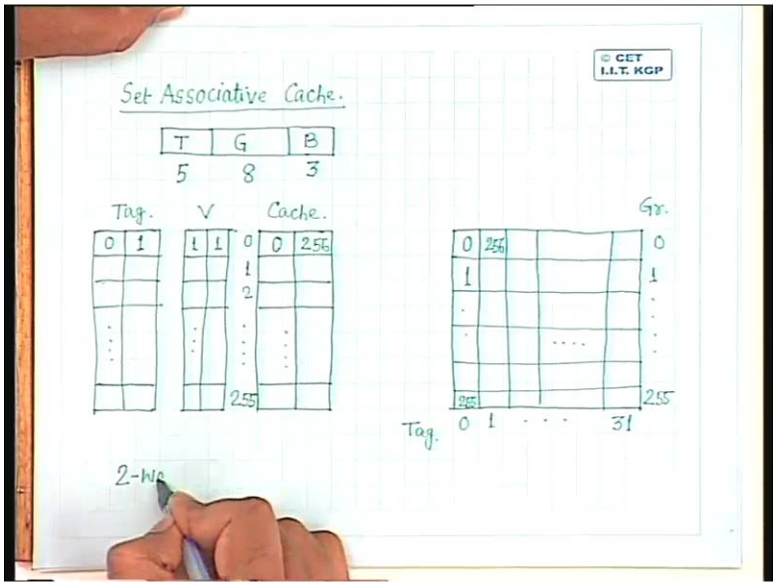
But obviously you have some disadvantages. In case of associative cache because you have flexibility that a main memory block can go to any of the cache lines, so the number of cash miss is less in case of associative cache. Whereas in case of direct mapped cache because a particular block of main memory can decide only in a particular cache line and it may so happen because I have only one option, I don’t have any choice. So it is quite likely that whenever I want to bring in a new block from the main memory and want to place it into the corresponding cache line, the cache line may be filled up by some other block of the main memory belonging to the same group of blocks.

(Refer Slide Time: 0:02:53.6 min)



So the number of cache miss in case of associative cache is obviously higher than that in case of direct mapped cache is obviously higher than that in case of the associative cache. So a solution which tries to incorporate the advantages of both that is we want to reduce the hardware at the same time we also want to maintain some flexibility so that instead of having only one cache line where a block can go, I want to keep some options may be out of 2, 3 cache lines, 1 cache line can be filled up by particular block from the main memory. So effectively what we are trying to do is we are trying to reduce the hardware as is done in case of direct mapped cache. We also try to take the advantage of flexibility as in case of associative cache.

(Refer Slide Time: 00:04:40 min)



So we come to an architecture of the cache memory which is called set associative cache. So the architecture is known as set associative cache. So even in case of set associative cache, the main memory address which is generated by the CPU is again divided into three fields as in case of direct mapped cache that is we will have a byte identification, we will have a group number and we will have a tag field. So taking the similar example as we have done in case of direct mapped cache, suppose this byte identification requires 3 bits, group identification requires 8 bits and tag identification requires 5 bits, having the same 16 bit address, main memory address. So similarly as we have done in case of direct mapped cache, what we can do is we can break the main memory or arrange the main memory blocks in the form of a two dimensional array like this. So I will have a number of groups because there are 8 bits in the group identification field. So obviously I will have 256 number of groups, so these are the group identification 0 1 like this I have 255, so total 256 number of groups, so this is group identification.

Similarly I will have 32 tags numbering from 0 to 31. So as before, the block number 0 of the main memory belongs to group number 0, block number 1 in the main memory belongs to group number 1. Similarly block number 255 in the main memory belongs to group number 255, 256 belongs group number 0 but tag number 1. So it is the same arrangement of the main memory blocks that we have done in case of direct mapped cache. The difference is in case of direct mapped cache, we have said that a particular block of the main memory belonging to a particular group can go to only a particular cache line in the cache memory. In case of set associative cache, what we do is instead of saying that this can go to only one cache line, we identify a set of cache lines. So you say a set of cache lines may consist of say 2 cache lines, 3 cache lines, 4 cache lines and so on but normally it is 2, 4, 8 like that, as is done in binary number system.

So if a set consists of 2 cache lines then we say that any block belonging to a group can go to that particular set and since within the set we have 2 cache lines, so any of the 2 cache lines can be occupied a block belonging to a particular group. So following the same logic, what we do is now let us assume that in every set we have 2 cache lines. Since we have say 255 groups in the main memory, 255 groups of blocks, so let us assume that we will have 255 sets of cache lines and every set will consist of 2 cache lines. So these two belong to the same set, this is the cache line. Now I have 255, 256 sets and every set consist of 2 cache line. So as I have 2 cache lines in the same set, so accordingly in the tag memory organization, I also have to have similar organization that is the tag memories will also be divided into a number of sets where every set will consist of two tag memories. So let us put it this way. So these are the tag memories which will also consist of 256 sets and every set will consist of 2 tag memories.

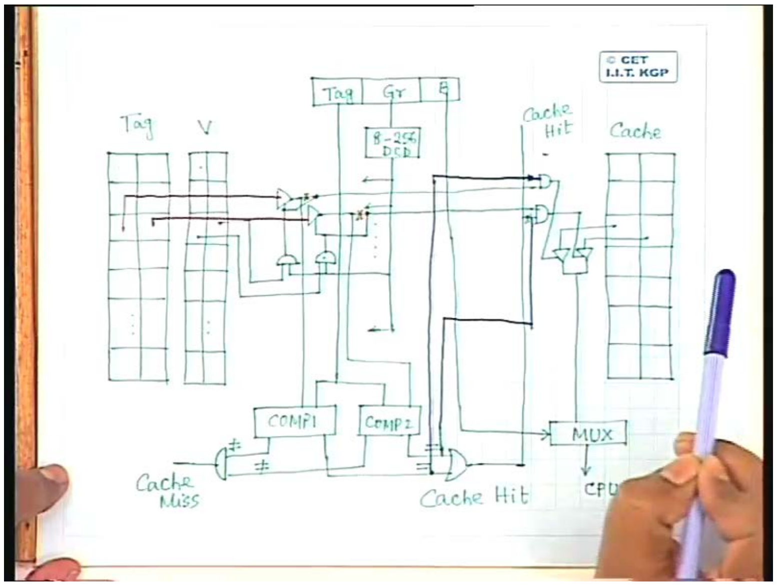
In the same way I will also have valid invalid bits. Now for every tag memory I have to have a valid invalid bit that means they will also be divided into similar sets, every set consisting of two valid invalid bits. [Conversation between the professor and student - Not audible (00:10:09 min)] In tag, for every cache I have to have a tag because cache will accommodate a block of the main memory and the tag memory will contain the block identification number. So now since we are saying that a group of any member of this group of a particular group will go to this particular set. Now this set consists of two cache lines. For every cache line I have to have a tag because otherwise I won't be able to identify that which block of the main memory is contained in which cache line. So that way if I decide say block number 256 will go to say this particular cache line.

For block number 256, what is the tag? Tag value is 1. So this entry, this particular tag memory will contain a value 1 and the corresponding valid bit will also be 1.

Now if I wanted to put another block belonging to the same group also in the cache memory, you find that the other entry of this set, other element of set is empty. So suppose this block number 0 has to be placed into the cache memory. In this case this particular entry in the cache, this particular cache line can contain block number 0. For block number 0, tag value is 0. So this corresponding entry in the tag memory will contain a value 0, this valid bit will be equal to 1. Unlike in case of direct mapped cache where we had to replace this block number 256 by block number 0 because there I didn't have any option. So here we get some amount of flexibility not as high as in case of associative memory but the restriction of the direct mapped cache is relaxed to some extent. [Conversation between the professor and student - Not audible (00:12:27 min)] Cache memory requirement is also doubled and since every set in this case consists of two cache lines, this is called a two way cache associative, two way set associative cache.

So similarly I can have 4 way set associative cache where every set will consist of 4 cache lines. Similarly I will have for every set that has to be 4 entries, 4 tag memory locations, I have to have 4 valid invalid bits. Now what will be the architecture for this type of set associative cache memory? It is simply the extension of direct mapped cache.

(Refer Slide Time: 00:13:25 min)



So here let us assume that as I said that even in this case that main memory address will be having 3 fields the byte field, the group field and the tag field. Using the same example since the group field contains 8 bits, so this will be fed to 8 to 256 decoder and depending upon the values in this group field one of the 256 outputs of this decoder will be high. On this side let me put the cache memories. Now the cache memory will be organized in the form of sets where every set will consist of two cache lines. Here let me put the valid invalid bits and the tag fields, so these are tag fields. In every set I have to have too entries in the tag field, I also have to have the valid invalid bits.

When I put it like this, it means that these two belong to same set, these two belong to same set, these two belong to same set. Now as before in case of direct mapped cache what we have seen is using the output of the decoder which is active which is enabled, I enable the corresponding tristated buffer to get the content of the tag memory location. Now in this case in case of set associative cache, you find that whenever a particular output of a particular decoder output is active that does not uniquely identify a particular tag memory location. It identifies a set of tag memory locations and in this case the set consists of two tag memory locations that means I have to compare both the tag memory locations to find out whether there is a cache hit or there is a cache miss. So suppose this particular output of this decoder is one, so when a particular decoder output is one that indicates that there may be 2 different tag memory locations which can give a hit. One of the two tag memory locations will be able to give a hit. So I have to compare 2 tag memory locations unlike in case of direct mapped cache where we have to compare only one tag memory location. So here the situation will be like this that using this same decoder output, I have to compare two tag memory locations.

So suppose let me see that this is the tag memory location that we want to compare. So the corresponding valid invalid bits will come here. So here also I will have two different buffers, one of the buffer input will come from this tag memory location. Let me use some other color, it will come from say this tag memory location and this one let us say comes from this tag memory location. The AND gate outputs, this AND gate output will activate this particular buffer, this AND gate output will activate this particular buffer. So you find that any of these can be valid and depending upon which one is valid, I have to compare the corresponding tag output. Now it may be possible that both of the bits are valid because this can contain a particular block, this can contain another block of the same group in which case both these valid bits will be equal to 1. so I have the possible tag values which are two in this case, out of this I have to select one that means in this case I need 2 comparators unlike in case of direct mapped cache where I needed only 1 comparator. So I have to have 2 comparators, this is comparator 1, this is comparator 2. So this output will go to one of the comparators, this output will come to another comparators. Both the tag values are to be compared with the tag value as given in the main memory address. Is it okay? Now I can have different combinations that is this tag value does not match with any of these tag values.

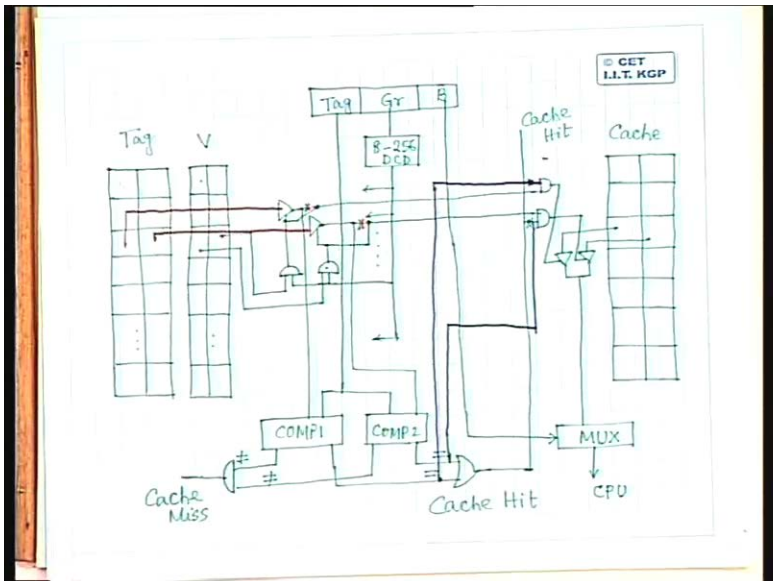
The tag value has given in the address does not match with the content of any of these tag memories. That means both the comparators give a signal of not equal. When such a situation arises this means the requested block does not exist in the cache memory because if it exists, it has to be present, the corresponding tag has to be present either in this tag memory or in this tag memory because it does not match, this tag value does not match with content of any of these tag memory locations that is an indication that the requested block is not available in the cache. So in this case when both the comparators signals not equal that is a condition which is the cache miss condition.

Similarly cache hit condition will be either this one indicates a cache hit or this one indicates a cache hit because the requested block, the tag value of the requested block may be present either in this or in this. So if it is present in this in that case this comparator output will be equal to comparator will signal equal. If it is present in this one, in that case this comparator will signal equal. So if any of them gives a cache hit that is the condition of cache hit, so I have to have an

OR logic. So following the same argument what I will do is this is the condition of cache hit. I will take one output from here; I will another output from here. They will be ANDed together, these two will also be ANDed together and they will activate two different buffers. So I am considering the third entry. [Conversation between the professor and student - Not audible (00:22:32)] Sorry, you are right. This will go to the AND gate outputs, not this.

So in the same manner, I will have two different buffers. These buffer will get input from this cache line, this buffer will get input from this cache line then this will activate this buffer, this will activate this buffer. These buffer outputs can be connected together, output of this comes to a multiplexer and the select input of the multiplexer comes from the byte identification line and this is the data that has to be returned to the CPU. So we find that in this case, we have slightly increased the hardware because in case of direct mapped cache we had only one comparator. Now we have two comparators but it is much less than that in case of associative memory where I had n number of comparators [conversation between the professor and student - Not audible (00:24:22)] yes, both AND gate outputs will be one. There is a point, you are right.

(Refer slide time 00:24:54 min)

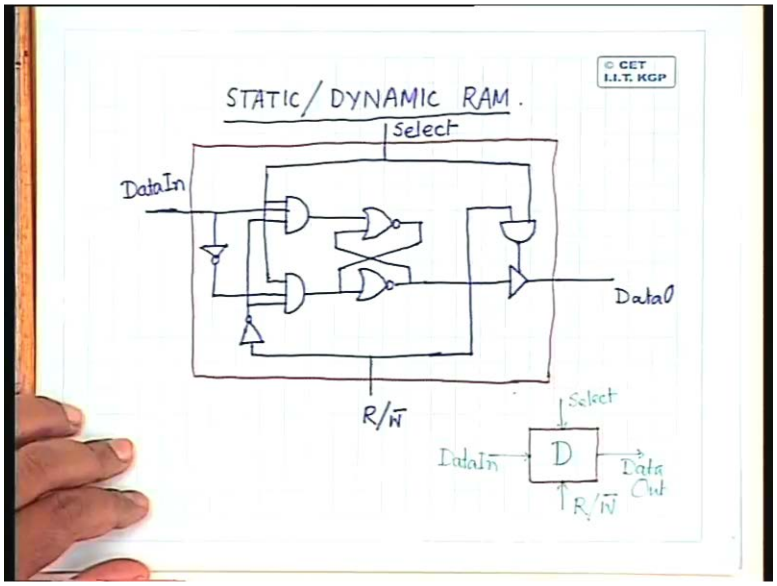


So what I have to do is instead of taking this cache set outputs, so this is a cache set signal. I have to take the outputs from here, this one will go to one of the AND gates. So instead of taking the connection from here, let me connect this here and the other one will go from this point. Is it? So this one will be off, I have to take this output and AND this one. Is it okay? That's right, thank you. [Conversation between the professor and student - Not audible (00:25:51)]. See as he has pointed out and it is right that both the tag memories can contain some valid tag value in which case both the valid bits will be equal to one. If both valid bits are 1 and I have a cache hit then if I AND these AND gate outputs with this cache hit signal, so both of these AND gate outputs will be equal to 1. If both of them are equal to 1, then both these buffers will be enabled. If we enable both the buffers then there will be data clash because I have directly connected the outputs of the buffers. Instead of that if you take the cache hit as given by the individual comparators, AND them with these AND gate outputs in that case one of these buffers will be enabled because both of the comparators cannot give output signal equal.

If both of them gives equal that will indicate that both of these cache lines contain the same data block which is not possible. So in that case only one of these buffers will be active and the data from the corresponding cache line will be available at the input of the multiplexer. So here you will find that we have been able to reduce the amount of hardware drastically compared to the number of amount of hardware that you need in case of associative cache, it is slightly more than that in case of direct mapped cache but we have been able to incorporate advantages of both. To some extent we have been able to maintain the associativity that is now a block of the main memory can go to one of these two cache lines.

Earlier in case of direct mapped cache, it had to go to only one cache line but now I have a choice. Any of these two, if free I can put the block in the corresponding cache line. After that concept can be extended instead of two way set associative, we can go for 4 way set associative, we can go for 8 way set associative in which case every set will consist of 4 cache lines or 8 cache lines and so on. Obviously in such cases, the number of comparators will also increase. If it is 4 way set associative in that case I have to use 4 comparators. If it is 8 way set associative, I have to use 8 comparators. So accordingly the scheme can be made. So after talking about the cache architecture and as we have just said that the cache memory is usually implemented with the help of static RAM whereas main memory is implemented with the help of dynamic RAM. So now let us see that what is the static RAM architecture or what is the dynamic RAM architecture?

(Refer Slide Time: 00:29:13 min)



So now we will consider about 2 types of RAM architectures static and dynamic. You might be knowing that every memory chip consists of n number of memory cells arranged in some fashion.

And typically a memory cell is nothing but some sort of a flip flop. Isn't it? So if I consider static RAM cell, the schematic diagram is static RAM cell can be drawn like this and we will find that it is nothing but a kind of D flip flop or latch. So the architecture of static RAM cell is something like this. So you find that whenever the read write signal this input is low, in that case whatever is available at the data in that will be latched in this particular latch and the latch consist of two nor gates. If we want to read anything from this latch or from this memory cell, the read write bus signal has to be high.

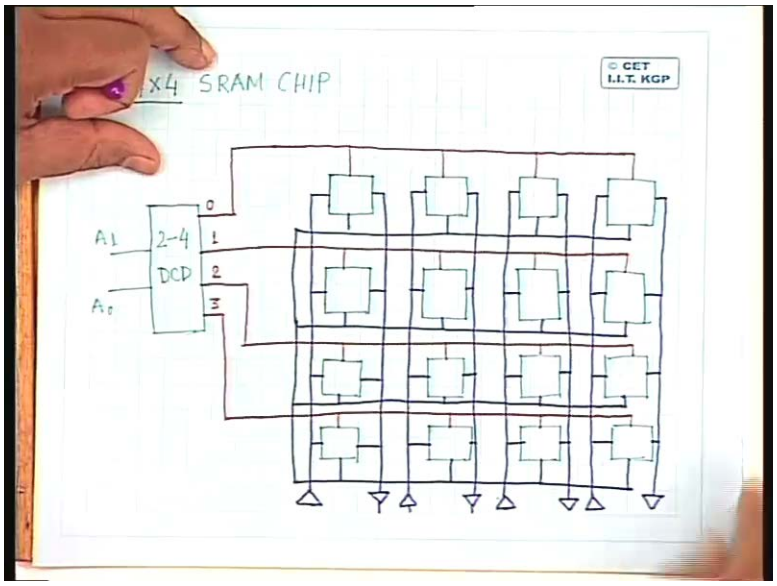
So if the read write bus signal is high and it is selected in that case this AND gate output will be high. When the AND gate output is high, this activates this particular buffer and when the buffer is activated, the output from latch will be available on the data outline. So this whole thing can be represented as a static RAM cell. So this is a static RAM cell, in other words I can put static RAM cell just by box like this which will have select input, a data inline, data outline and read write bar input and this is a static RAM cell D. Now if this static chip is nothing but an array of such cells arranged in some fashion, so usually these RAM cells are arranged in the form of a two dimensional array. It is something like this, say if I wanted to have a static RAM of say 4 locations, every location consisting of say 4 bits. So I will have a 4 by 4 SRAM cell or SRAM chip.

The organization will be something like this because I have 4 different locations and every location will consist of 4 bits that means for every location, I have to have 4 such static RAM cells placed in a linear order. So I can consider that this is a row of 4 such static RAM cells and since I will have 4 locations, each consisting of 4 bits, so I will have 4 such rows, in every row I will have 4 static RAM cells. Now to read or write a particular location in this RAM chip, I have to select the entire row. Suppose I wanted to read the data from the zeroth row in this static RAM chip, in that case all the cells in the zeroth row must be activated simultaneously. That means the select inputs of all RAM cells in the zeroth row are to be selected simultaneously. Then if I want to read the data, the read lines of all the RAM cells have to be activated simultaneously.

If I want to write data in that case this read write bar line has to become low simultaneously and the data must be available to all the RAM cells simultaneously. So I can arrange... [Conversation between the professor and student - Not audible (00:36:26)] this is just a flip flop. I think you have done flip flop in your digital course. This is just a flip flop so whatever data you feed to the flip flop that gets latched. How do you latch it? You feed some data to the data input line. When you feed the data into the data input line, if at the same time your select input is active and I want to write the data that means read write bar signal has to be low. Here you find that this read write bar signal goes through an invertor to both these AND gates. So if this is low, this inverter output will be high that means this AND gates will be active, will be selected provided the select input is high and read write bar signal is low. In such case whatever you feed to the data, data input line if this is say one in that case this AND gate output will be 1, second AND gate output will be 0.

So if this is one, you find that this nor gate output will be zero, this nor gate output will be one and it will remain until and unless you change this by feeding another data while that unit is selected and read write bar signal was low. That means by making read write bar signal low and selecting this particular unit, the memory cell you are reading the data into this memory cell, you are writing the data into this memory cell. Now if I want to read it, so read is clear. So every memory chip will consist of such cells arranged in some fashion. So here we are considering a two dimensional array of such memory cells and I am considering a 4 by 4 array.

(Refer Slide Time 00:38:38 min)



So let us put those cells, arrange those cells in the form of a 4 by 4 array. You remember that each of cells will have those 4 signals select input, read write bar input, data in input and data out output. Now whenever we select this memory for read or write then a particular row, all the cells in a particular row or all the cells in a particular row has to be selected. So what we do is let us connect the selects inputs of all the cells to a common line. So these are the select inputs.

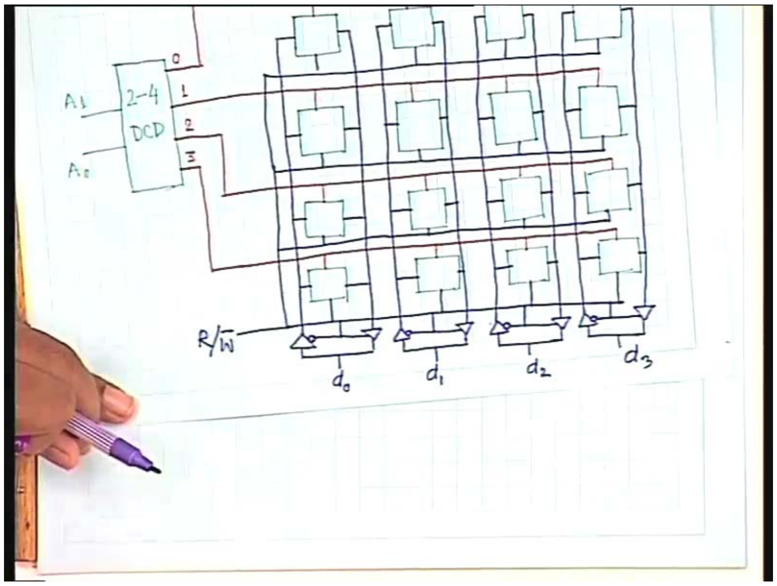
Similarly when we want to read any particular memory location or we want to write into any particular memory location in that case, the corresponding read write bar signal has to reach all the cells in the selected rows simultaneously or even if the read write bar signal reaches all the cells, whether the row is selected or not that does not matter because only the row which is selected that will be affected, other rows will not be affected. So what we can do is let us have a common read write bar signal which is connected to all the cells. So this is your read write bar signal and the red lines are the select inputs of individual rows.

Now since we have 4 different locations that means for addressing any particular location, we need to do address lines. And what we can have is just a 2 to 4 decoder, say if I have a simple 2 to 4 decoder which will have input lines A_0 and A_1, these are the address lines. Different outputs of this decoder can be used to select different rows in this memory array. It's very simple. So this is say zeroth output, first output, second output and third output. So you find that this particular array gets an address 0, these row gets an address 0, these row gets an address 1, these row gets an address 2 and these two gets an address 3. Now what is left is how to connect the data in and data out lines.

Again what we can do is very simple, say these are data in lines. we connect the data in lines of all the cells in a particular column together. Similarly data outlines of all the cells in a given column together. These are the data in lines. So let me put a buffer in the inward direction, this is data outline, so let us put a buffer in the outward direction. Now this buffer pair is feeding the same column, so let us connect these two together. Similarly these buffer pair is connected to the same column. Let us connect these two together. This is feeding the same pair, so let me connect these two together. This is feeding the same pair, so let me connect these two together. Now find that if the operation to be performed is a write operation that this is actually the external data lines. These are external data lines let me name them as d_0, d_1, d_2 and d_3. If I have to perform a write operation in that case d_0 has to reach inputs of all these cells and this is my read write bar signal.

So if this read write bar signal is low, in that case this buffer has to be active. If it is high then these buffer has to be active. So when you find that the read write bar signal is low in that case operation that we have to perform is data write operation. So whatever is the available on the external data line that has to be written into the cell which is selected by the select line. That means the data flow is in the inward direction, so this particular buffer has to be active. If it is a read operation in that case data from the selected cell has to come to the external database, external data line. So this particular buffer has to be active and that is set by this read write bar line. So I will have single connection to all these pairs like this and this becomes a 4 by 4 SRAM chip.

(Refer Slide Time 00:47:08 min)



So you find that whenever we wants to design a chip, we have to arrange these memory cells, the static RAM cells in the form of an array like this where every row will be selected by one of the decoder outputs and whether it is read operation or write operation that we want to perform that has be decided by the read write bar signal. So with this we stop now.