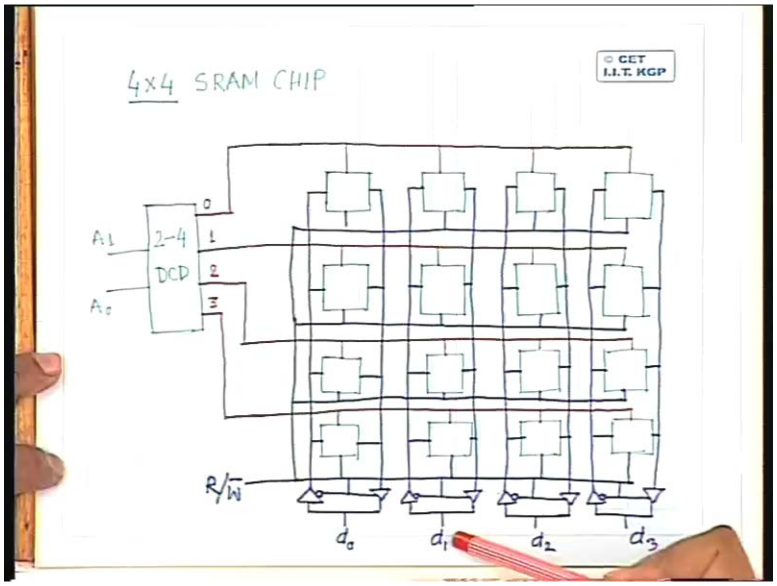
Digital Computer Organization

Prof. P.K.Biswas
Department of Electronic and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture No. # 20
RAM Architecture

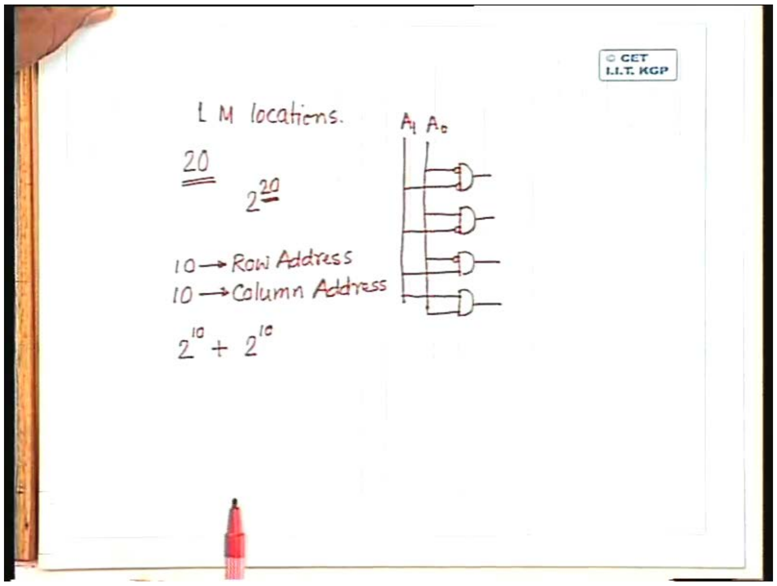
So though this architecture, two-dimensional array of RAM cells this appears to be okay logically but practically there is some implementation problem.

(Refer Slide Time: 00:01:11 min)



The problem is like this, if we have only one decoder like this which selects one of those rows, all the cells in a particular row then we will consider a situation so suppose I wanted to have a memory which will have say one mega locations. So one mega locations means I have to have 20 address lines. Now every location may consist of say one memory cell, it may consist of 2 memory cells, 3 memory cells and so on. But what I need is I have to have 20 address lines to address different locations or different rows in the cell array. So if I have this 20 address lines then you consider what is the decoder complexity. In the decoder I need 2 to the power 20 number of gates, AND gates because the decoder logic is simply something like this. It is nothing but an array of AND gates. So if I want to have simply a 2 to 4 decoder which will have 2 lines say A_0 and A_1, the connection will be something like this. So when both A_0 and A_1 they are 0 0 in that case this output will be high. For 0 1 this output will be high, for 1 0 this output will be high, for 1 1 this output will be high.

(Refer Slide Time: 00:01:25 min)

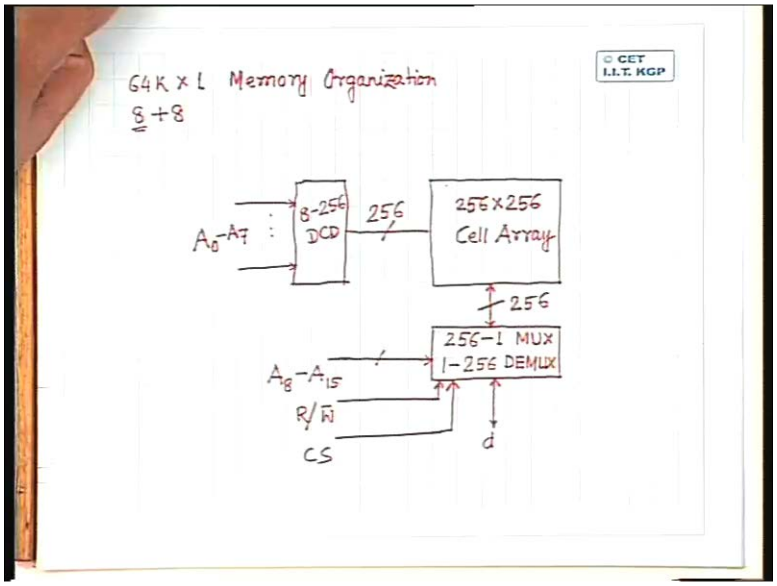


So for a 20 input decoder I have to have 2 to the power 20 number of AND gates. Every AND gate has to have a fan-in of 20. Isn't it? 20. So just like this, in this case every AND gate has 2 input lines for a 2 to the power 20, 20 to 2 to the power 20 decoder, I have 2 to the power 20 number of AND gates, every AND gate will have to have 20 input lines that is quite difficult. So instead of doing this, if we break the address in to two components, so this 20 address lines I break in to 2 groups, 10 10 each and I say that out of these 20 address lines, I will consider as row address and these two 10 address lines I will call as column address. And accordingly I will have 2 decoders, one of them will be called as row address decoder other one will be called as column address decoder. So the decoder complexity will come down drastically. For row address decoder I need 2 to the power of 10 number of AND gates.

Each AND gate will have 10 input lines. For this I need again 2 to the power 10 number of AND gates which will again need AND gates of 10 inputs each. So the number of AND gates that you need is 2 to the power 10 plus 2 to the power 10 that is 2 to the power 12, 2 to the power 11 whereas in the original configuration the number of AND gates was 2 to the power 20. So number of AND gates has been reduced to a great extent not only that the fan-in of every AND gate has also been halfed.

So instead of directly trying for only one decoder, for decoding all the address lines you break the address line in to two components, one of them you call as row address and the other set you call as column address. Accordingly you have two decoders, one is the row address decoder other one is the column address decoder and by making use of these two decoders, I can select different cells in the two dimensional cell array.

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



So using this if I want to implement a memory, let us take a memory of say 64 K where every location will contain just one bit. So I want to have a 64 K by 1 memory organization. Following this architecture it is 64 K means I have to have 16 address lines. So if I follow this architecture, I need a decoder which will decode 16 to 2 to the power 16 outputs, for 16 inputs it will give me 2 to the power 16 outputs that means I need 2 to the power of 16 number of AND gates, every AND gate will have a fan-in of 16. Instead of doing this I will break this 16 address lines in to 2 groups, each of 8 bits. So I will break it in to 8 plus 8 address lines. One group of 8 address lines I will call as row address and the other group of 8 address lines I will call as column address.

The cell, memory cells will again be arranged in the form of a two-dimensional array and it will be 2 to the power 8 by 2 to the power 8 that means 256 by 256 cell array. That means every row will have 256 cells and I will have 256 number of such cells. The 8 address lines which I call as row address that is fed to a 8 to 2 to the power 8 decoder so 8 to 256 decoder. It will have 8 address lines as input, so let me mark them as address lines A_0 to A_7. The decoder have 256 output lines, depending upon the bit combination of this 8 inputs, one of this 256 output lines will be active and the corresponding row in this two-dimensional cell array will be selected.

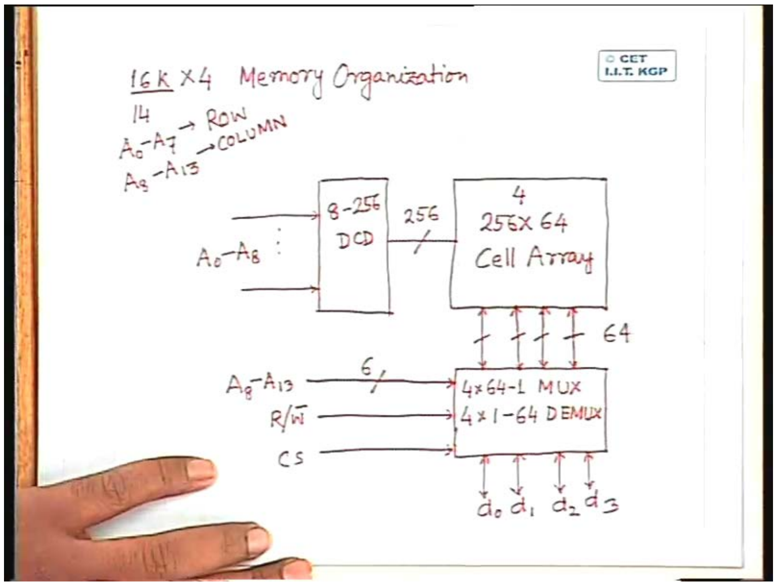
So when a row is selected that means all the cells in that particular say, in that particular row will be available either for reading operation or for writing operation. Now out of this 256 cells belonging to a particular row, I have to select one particular cell because my organization is 256 K by 1 memory organization. So when I am selecting a particular row either for reading operation or for writing operation, I am selecting all the 256 cells belonging to a particular row with the help of this row address decoder. Next what I have to do is out of those 256 cells, I have to select a particular cell for reading or writing operation and which cell I have to select that will be dictated by the parameters that is again another 8 bits. So the arrangement in this case that can be made is instead of simply a decoder, I can have a multiplexer demultiplexer combination. So if I use, if I want to perform a read operation of a particular bit, I pass all this 256 cell outputs through a 256 to 1 multiplexer.

If I want to select a particular cell for a write operation, at the output of this I have a single data line. So this is line d both for reading operation and writing operation assuming that external data bus that we have is a bi-directional data bus. So for reading operation out of this 256 cells, output of one of them will be available on the output data bus. For writing operation the data which is fed to this line has to reach one out of this 256 cells. So for reading operation I need a 256 to 1 multiplexer. For writing operation I need just the reverse 1 to 256 demultiplexer and these demultiplexer select lines actually come from the column address. So here I will have 8 lines, 8 address lines named say A_8 to A_{15} between the cell array and this multiplexer demultiplexer pair I have 256 lines.

Now in addition to this address lines, I also have to have the control signals like read write bar. I also have to have another control signal which we will call as chief select. So here what we are doing is using this row address which is 8 bits, I am selecting one of the rows in this two-dimensional cell array. A row means 256 number of cells. When I want to read a particular location that means I have to read a particular bit, a particular cell in that 256 cells so which cell I want to read that depends upon what address is being fed to A_8 to A_{15} lines depending upon that one of this 256 will be selected and will be available at the output of the multiplexer.

Similarly for writing operation I am selecting an entire row but the data will reach only that particular cell which is fed by this 1 to 256 demultiplexer. Again the selection of a particular cell through this demultiplexer will be done by this column address lines. So you find that I again tremendously if I break all the address lines in to two components row address and the column address. So this is the organization of it is a 64 K by 1 memory chip.

(Refer Slide Time: 00:13:56 min)



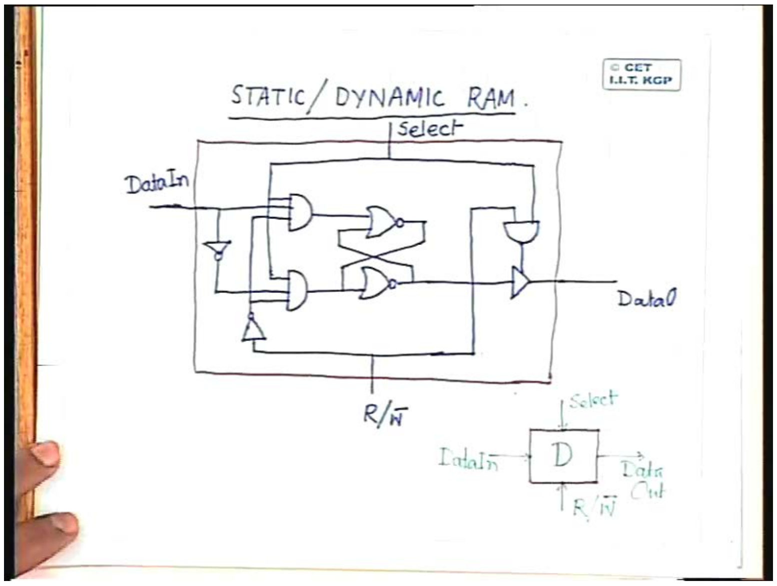
16K x4 Memory Organization

The same cell array can also be thought of like this, I can with slight modification, I can make it as a 16 K by 4 memory organization. You find that the number of cells are same. In this case also I have 64 K cells, in the earlier case also we had 64 K cells but the cell organization in this case will be slightly different. Here all the 64 K cells we can put into 4 different arrays, so I will have 4 256 by 64 cell array. Instead of a single cell array, now we use 4 two-dimensional cell arrays, every cell array will have 256 rows, every row will consist of 64 cells. Now because I have 16 K locations that was number of bits needed to address this 16 K locations is 14 because 2 to the power 14 is 16 K. These 14 lines I break again in the form of row address and column address. I will assume that 8 of these address lines A_0 to A_7 will be the row address and the remaining 6 address lines A_8 to A_{13} will be the column address.

So as before because I have 8 address lines identified as row address, so I have to have decoder that is 8 to 256 decoder. These decoders will have 8 address line inputs A_0 to A_8, it will give 256 output lines, one of them will be active at a time depending upon this bit combination on the input side. So whichever output line is active, the corresponding rows because now I have 4 such cell arrays, so it will select 4 rows, one from each of the array. In the earlier case we had only one multiplexer and demultiplexer but now because we have 4 such arrays so I need 4 64 to 1 multiplexer, one multiplexer for each of the cell array. I also need 4 1 to 64 demultiplexer. I will have 4 input output lines, one from each of the cell array let me mark them as d_0, d_1, d_2 and d_3. These input, this unit the multiplexer demultiplexer unit will get the column address which consists of 6 lines and the addresses from A_8 to A_{13}. It will have control inputs read write bar, it will also have control inputs a chip select and now here I need 4 such connections each of width 64.

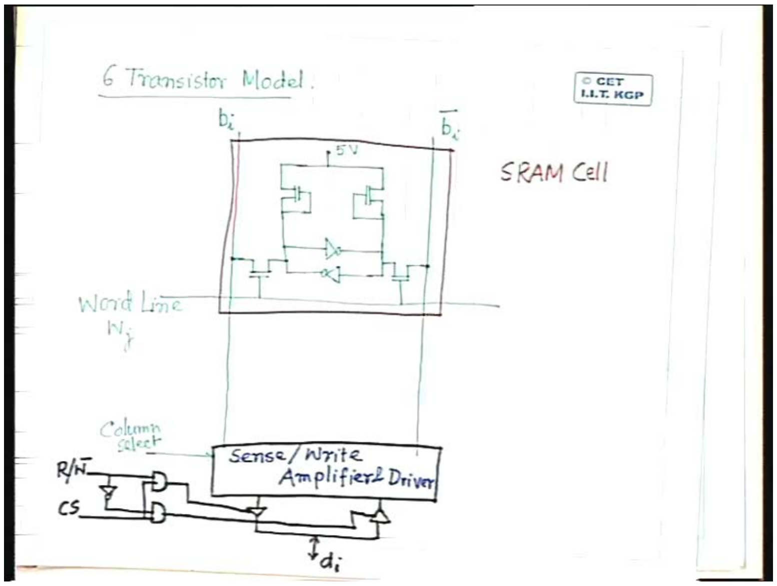
So you find that by making use of various organizations, various kinds of cheap arrays, I can go for different kinds of memory organization but the basic structure will be like this. I have to have a row decoder, I have to have a column decoder. That row decoder will select an entire row consisting of a number of memory cells and the column decoder will select one or more cells depending upon what is the memory organization from the cells coming from a particular row. Is it okay? Now let us see practically how these cells are implemented.

(Refer Slide Time: 00:19:35 min)



You find that as we have said in the cell architecture, the basic storage element is only this which stores the bit either 0 or 1 but I have a number of additional gates for every cell and the number of gates which are used to control the cell is much more than the number of gates which are actually used to store the value. That means the control overhead is much more than the storage overhead. Naturally this is not an architecture which is desirable. So for implementation and particularly many of you have become expert in VLSI by this time, so particularly for VLSI implementation, we have to keep in mind that the number of components should be reduced as much as possible.

(Refer Slide Time: 00:20:38 min)



So for practical implementation each of these cells is implemented by using a model which is called a 6 transistor model. And it is very simple; every cell will be something like this. So it is something like this. This is the particular cell (Refer Slide Time: 25:13). Now we find that this cell consists for 6 transistors because each of this inverters can be implemented using a single transistor. Now the cell is, all the cells belonging to a particular row is selected by this word line W_j which actually comes from the row address decoder and using the column select, from the column address decoder actually there are a number of columns like this, if I repeat this same architecture one after another this becomes a number of columns and the rows will consist of these cells arranged one over the other.

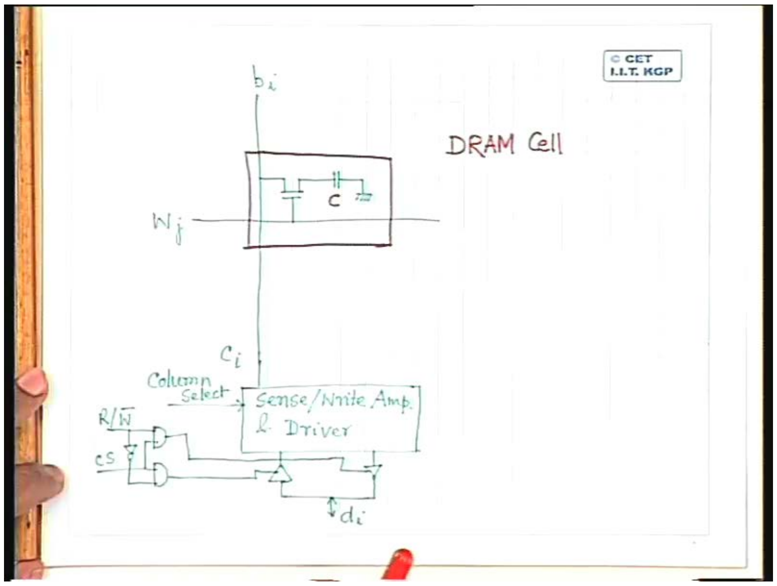
Now see that how do you write the data in to this cell or how do you read the data from this cell. Whenever we want to write a data say data one in this cell, what we have to do is we have to drive this b_i line these are this is called a dual rail. I have to drive b_i line to high and b_i complement line to low. So with b_i line to high and b_i complement line to low, if you select this particular word line in that case one will be written in this latch. You will find that this is nothing but a latch. So what is the difference? The difference is the NOR gate has been replaced by an inverter. So, if I set this line to high and this line to low and select this particular cell through this word line W_j in that case one will be written in this side.

Similarly if we want to write a 0, I have to drive the b_i line to 0, b_i complement line to high or 1 and select this cell through this word line W_j, in that case 0 will be written in the cell. At the input you are giving only a single bit either 0 or 1 so that bit has to be converted to a complement pair. If I give an input one that has to be converted to 1 and 0, b_i line has to be 1 and b_i complement line has to be 0 so which is to be done by this unit. This is actually sense write amplifier along with that this line also performs the additional task of driving lines b_i and b_i complement both for reading purpose and writing purpose. So I can put it, it is sense write amplifier cum driver.

Similarly if I want to read this particular memory cell in that case what I have to do is we have drive both b_i and b_i hat lines, b_i complement lines halfway between 0 and 1 logic. So if the zero logic is voltage 0 and 1 logic is say voltage 5, 5 volts then for reading purpose I have to drive both the lines to something like say 2 volts or 2.5 volts. So driving these two lines b_i and b_i complement to a voltage in between 0 and 1 logic level and then selecting this cell through this word line W_j, we will find that then whatever is stored in this particular cell these lines b_i and b_i complement will be taken accordingly. So that is sensed by this sense amplifier and the corresponding output comes to this output line d_i.

So if we implement the cell like this, in that case you find that the number of logic, number of transistors that is required is much less compared to the number of transistors that is required in this type of implementation. So this is the basic architecture of static RAM cells and static memory organization. But as we said that this static RAMs are mainly used for implementation of cache memory whereas whenever we go for implementation of the main memory, we don't use static RAMs rather we use dynamic RAMs. So lets us see what is the organization of a dynamic RAM. In case of dynamic RAM, the RAM cells are very simple, it is simply like this.

(Refer Slide Time: 00:30:15 min)



So this is word line W_j and this is for a particular column say C_i. The other part that is the write sense amplifier and driver which is almost similar to that in case of static RAM cell. So this is sense write amplifier and driver and as before I will have a number of buffers, this is connected to output data line say d_i. I have the other control circuitry and here I have column select. So dual rail in case of static memory cell is replaced by a single rail which is b_i in this particular case and this is the basic cell architecture of dynamic RAM, so this is a dynamic RAM cell. So in this particular case a bit one is stored in this cell whenever this capacitor is charged to high voltage. If the capacitor is discharged that means the bit that is stored in this cell is 0, I don't have any latch.

So if I want to store a particular bit say 1, what I have to do is this b_i line, this single rail b_i has to be driven to a high voltage and after that you have to select this particular cell through this W_j line and since b_i is driven to a high voltage and this cell is selected that means the path from this b_i line to this capacitor is on, this is the capacitor in which case the capacitor will be charged and it will store a value 1. If I want to store a value 0, what I will do is I will set this b_i line to low after that I will select this cell through this W_j line, so if the capacitor was charged previously, this will now be discharged, so it will store a value 0. Now if you compare the circuit complexity of the dynamic RAM cell with the static RAM cell, this is static RAM cell, so obviously you find that the number of components in the dynamic RAM cell in the DRAM cell is much less than that case of SRAM cell. So that is why the packing density of the DRAMs is much higher than that in case of SRAMs and that leads to cost reduction in case of DRAM. But simultaneously I have other problem. In case of SRAM even if you read a value from a RAM cell, the value contained in the RAM cell is not changed because the value is already latched.

In case of dynamic RAM cell, you assume that the value stored in this cell was one that means the capacitor was charged. Whenever I want to read this cell what I have to do is I have to select this cell through this line W_j then the cell will be selected, the value stored in the capacitor, the charge stored in the capacitor will be sensed by the b_i line. The moment you try to sense it, this capacitor will get discharged that means whenever I want to read a value, the capacitor will be discharged. So this is what is called a destructive read out. So I have to have an arrangement that because I know that whenever I read a RAM cell, if it is storing a value 0 then there is no problem but the problem comes if it is storing a value 1.

So whenever I want to read a RAM cell and the RAM cell was containing a value 1, the capacitor is going to be discharged and the value is going to be lost. So I have to have an additional facility that whenever you read any RAM cell, I should be able to refresh the RAM cell back that means whatever value is there, the immediate next action should be you write the same value in the same RAM cell which is called DRAM refreshing that means after every read operation the DRAM has to be refreshed. Not only that if I leave this RAM cell just like this without any reading operation, even then the capacitor will be discharged because of leakage. So even if I don't read the DRAM location, I should refresh the DRAM at regular intervals of time. That means I have to have a mechanism by which the DRAMs can be refreshed at regular interval of time or it will be refreshed after every read operation. So details of all these DRAM architecture we will see in the next time.