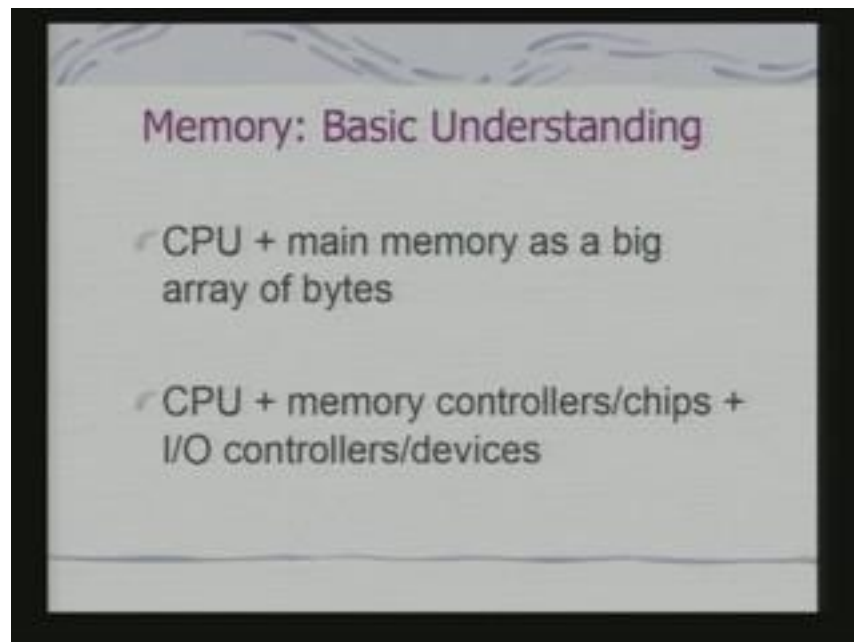


**Embedded Systems**  
Dr Santanu Chaudhury  
Department of Electrical Engineering  
IIT Delhi  
Lecture 11  
**Memory**

We have found in our earlier discussions that memory is a very important component in Embedded Systems. Today, we shall look at different types of memory and how they can be used. Let us recapitulate the concept about the memory and its usage in a, in an Embedded System as well as in a general computer. So, we can view that CPU and the main memory has a big array of bytes and that is basically is my micro- controller or a computer.

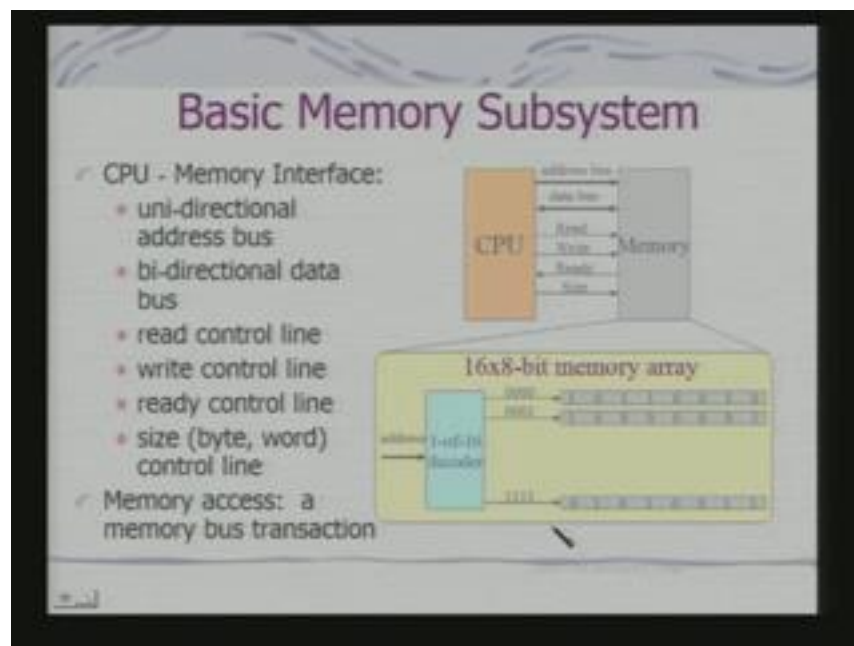
(Refer Slide Time 02:06 Min)



We can also put in, and in between memory controller to interface with external I/O devices as well as memory. In fact we have seen a number of these processors or processing codes targeted for Embedded System have memory interface units provided for this purpose. So, the basic structure or the organization is something like this; I have

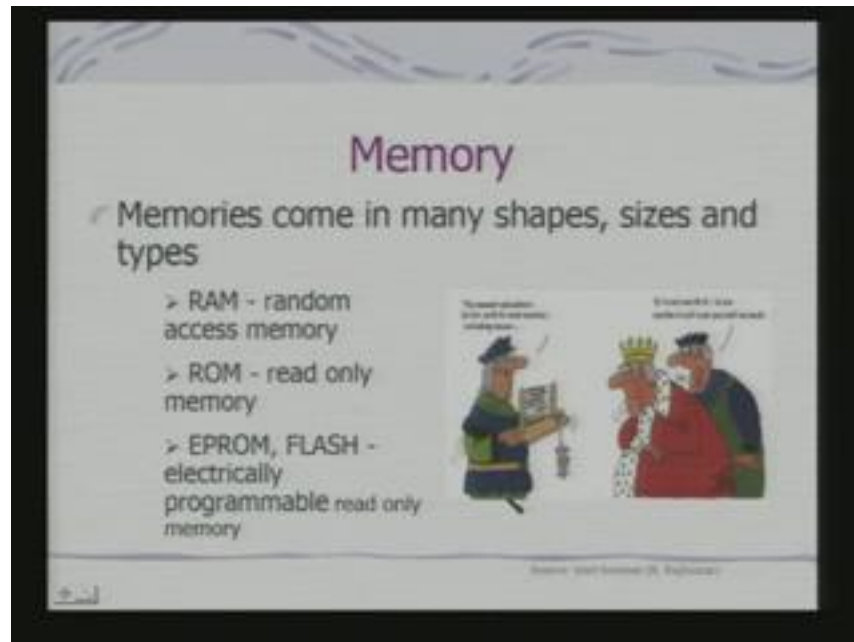
got the CPU, CPU is interfaced with directly here with the memory, there will be an address bus to select the memory location, bi-directional data bus for sending the data to write on to the memory as well as to get data from memory for the purpose of reading. These are the control signals- read, write as well as this is a ready signal if the memory is not being able to response to the request it may ask the CPU to wait so that is why another interfacing signal is a ready signal. The size indicates the number of pipes that are expected to be transferred through one read or write cycle. So, effectively a memory access, what we call is a bus transaction and the bus transaction or the nature of the bus transaction can be many cases defined by a finite state machine. So, here we are showing how this decoding can take place, there would be address it will be decoded and individual words in the memory will be selected.

(Refer Slide Time 03:57 Min)



Memories come in many shapes and sizes and the technology changes. This is a cartoon which says that today I can have a technology with certain features but tomorrow it may change drastically and this is true with memory, with continuous evolution in the technologies that are going into their manufacturing.

(Refer Slide Time on 04:07 Min)



**Memory**

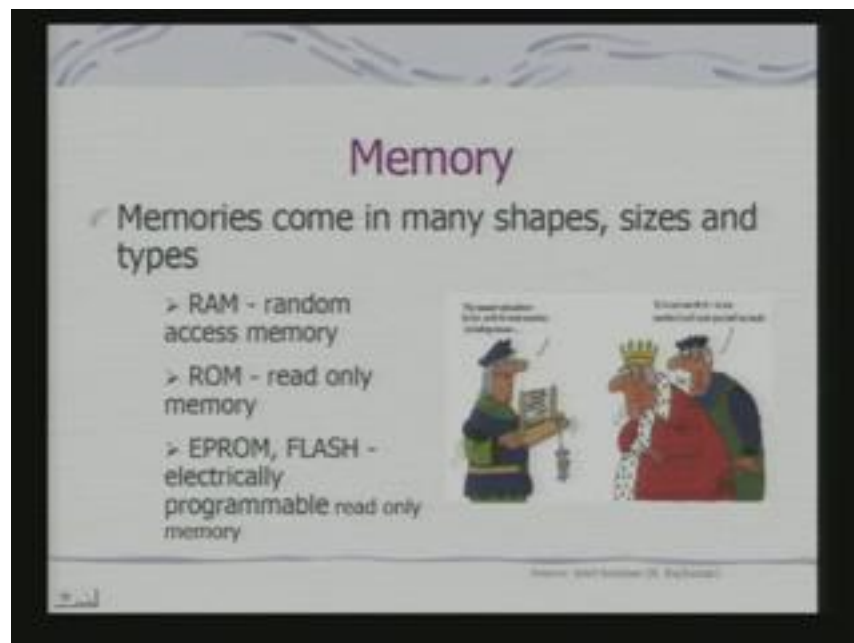
Memories come in many shapes, sizes and types

- > RAM - random access memory
- > ROM - read only memory
- > EPROM, FLASH - electrically programmable read only memory

The cartoon illustration shows a man in a blue uniform holding a document and talking to a man in a red robe and crown. The man in the blue uniform says, 'The computer system is in and it's ready to go.' The man in the red robe says, 'It's wonderful to see you here and we'll be sure to get it done.'

Basically, we have this brought categorization- RAM random access memory, ROM read only memory, EPROM or FLASH electrically programmable read only memory, but it is interesting to note that ROM as well as EPROM and FLASH. They enable random access that means you can access any location in a memory and you did not access them serially or sequentially.

(Refer Slide Time 04:47 Min)



**Memory**

Memories come in many shapes, sizes and types

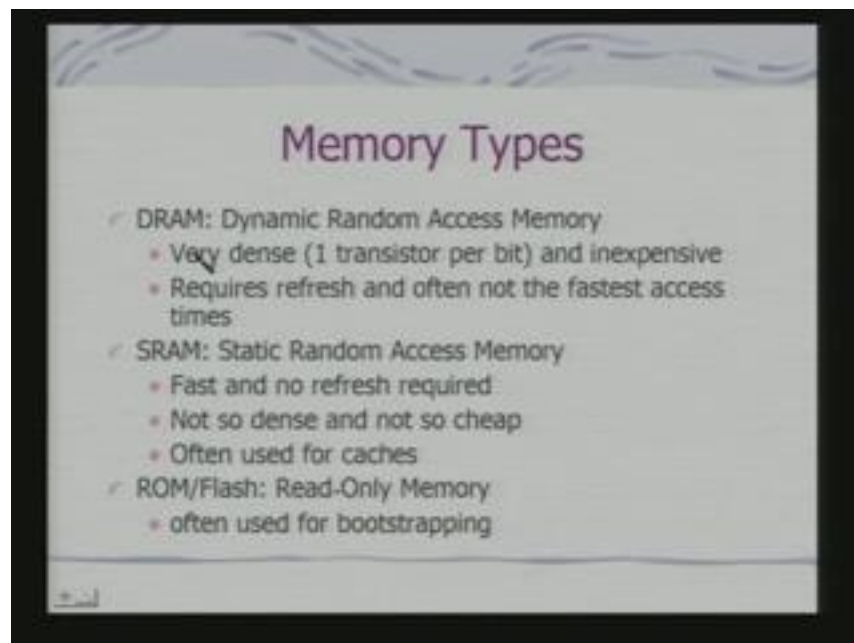
- > RAM - random access memory
- > ROM - read only memory
- > EPROM, FLASH - electrically programmable read only memory

The cartoon illustration shows a man in a blue uniform holding a document and talking to a man in a red robe and crown. The man in the blue uniform says, 'The computer system is in and it's ready to go.' The man in the red robe says, 'It's wonderful to see you here and we'll be sure to get it done.'

Further we can look at their divisions into dynamic RAM- DRAM, static RAM that is SRAM, ROM or FLASH which are read only memory and read only is a, of a news for bootstrapping, in fact FLASH has got some characteristics so that we can even do in circuit writing as well. So, they are not strictly read only but their primary usage in applications may be for read only purposes. Here we have got a comparison of their characteristics; in fact dynamic random access memory is a very dense and consequently inexpensive, but it requires refresh. Why it requires refresh? We shall understand when we look at its structure. Static Random Access Memory is first, no refresh is required but it is not as dense as that of Dynamic Random Access Memory and often it is used for cache or other applications where we require really fast memory.

What do we mean by density of the memory? It is actually the area that is required on the silicon real estate for implementation of a single bit. When we see DRAM is more dense than that of SRAM we mean that the area required for a implementation of a single bit of DRAM is much less.

(Refer Slide Time 06:30 Min)



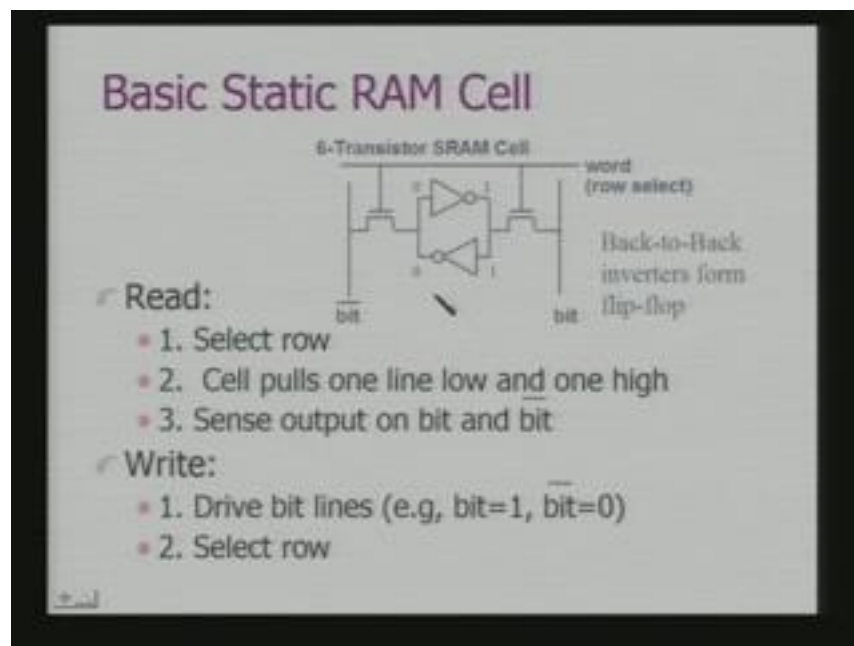
Let us first look at static RAM; a typical static RAM cell that is a single bit implementation is something like this. It requires about 6 transistors for its implementation. You can understand that this back to back inverter actually forms a Flip-

flop, so the bit is retain in terms of the states of this Flip-flop. So, when you read a data, what you do? You select the row, so this is your select word or row select light. You select the row selecting the row means effectively you are activating these transistors and you sense this bit bar and bit line, okay. This, actually these two lines are actually fed to what we call a sense amplifier, so the difference **aim** the occurrence is actually sense to read the data.

So on the other hand, how do you write? You drive bit lines and select row and accordingly what you do, you force the Flip-flop to change state if it is required.

So this is how a bit gets stored in a static RAM; obviously since it requires 6 transistors to implement a single cell the area required for implementation of the single bit SRAM is more.

(Refer Slide Time 08:10 Min)

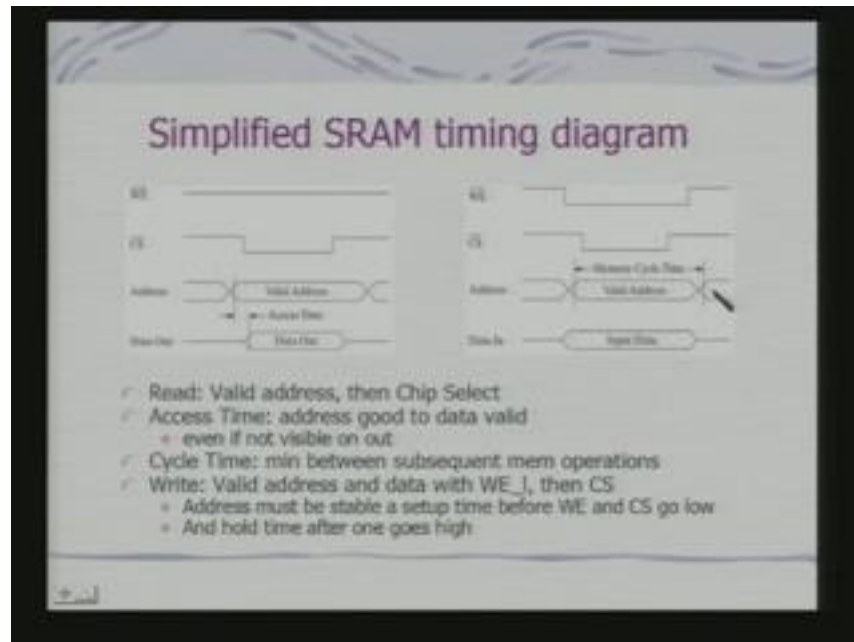


This is a simplified SRAM timing diagram that means how the signals should be sent or received from the static RAM for the purpose of reading and writing. So, here we are using a very simplified form where we have got this right enable. So, when it is, it is an active low signal, when it is high that means we are basically doing a read. The chip select is basically selecting the corresponding IC or the memory module for the purpose of reading. The address is provided on the bus, the decoder would be typically part of the

memory device itself for selecting the row. So, that means once I have selected the corresponding memory module and I have provided the valid address, the row would be selected and after that you will be expecting the data. So, this is the time difference between make, between the address becoming valid and receiving and you receiving the valid data. So, read is valid address than chip select, access time address good to valid data even if it is not visible on out that means it may take sometime to get it visible because there may be output buffers. So, this is basically a critical parameter to estimate performance of a particular memory block that we are using. Access time smaller means the memory is faster.

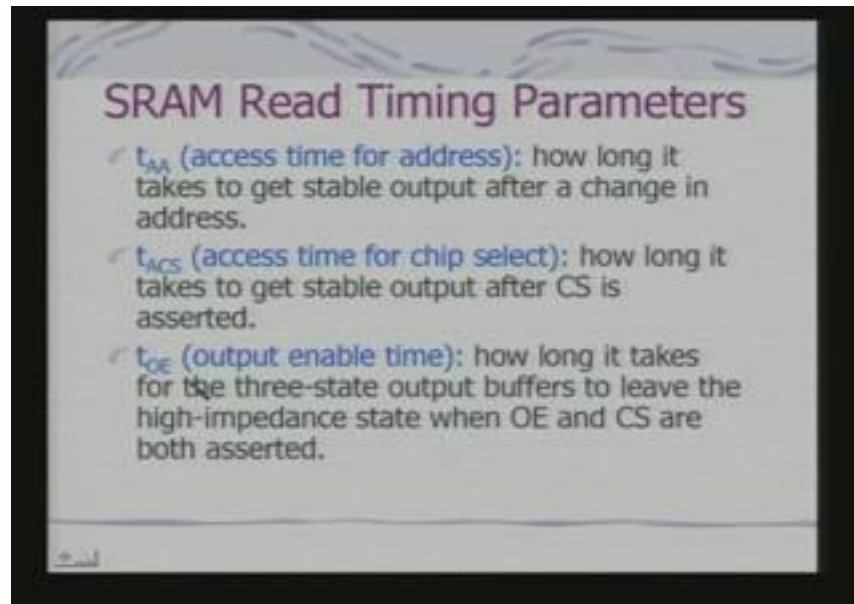
Now, in this case what we are showing is, you are making basically right. So, valid address is there, data is also valid your, your write enable is validated and then you have got the chip select and then you actually write the data. So, what it says that address must be stable as set of time before write enable and CS goes low because address has to be valid for the purpose of selection. So, there has to be a set of time between the write enable and CS to go low and hold time after it goes high. So, this is basically the hold time. So, what we say this is the memory cycle time. The minimum time between subsequent memory operations because I cannot have a new address put in; I cannot have a new address put in before expiry of a memory cycle time if I am doing a write or a read operation.

(Refer Slide Time 11:00 Min)



Now, let us look this timing in more detail. So, this is the typical scenario where I have also assumed that there is at the output stage and buffer, a 3 state buffer. So, one of the state is a tri state high impedance state and I have got an output enable signal which enables the output buffer. So, here we have got more timing parameters which I have shown in this context. So, what you have got here is, you have got the address which is shown here as a basic stable part and here we are showing the data which is coming out as a valid data because I am now looking at primarily the read timing, okay and this output enable. So, when there is output enable low you actually are getting the valid data. We shall now look at specific definitions of these different parameters. So,  $t_{AA}$  is called access time for address- how long it takes to get stable output after a change in address. Next is access time for chip select- how long it takes to get stable output after the CS is asserted.  $t_{OE}$  it is the output enable time- how long it takes for the 3 state output buffers to leave the high impedance state when OE and CS are both asserted. So, you have found that I have now refined the definition of access time. The access time definition now we are providing with respect to address as well as with respect to chip select and a parameter is being defined with respect to the output enable signal as well.

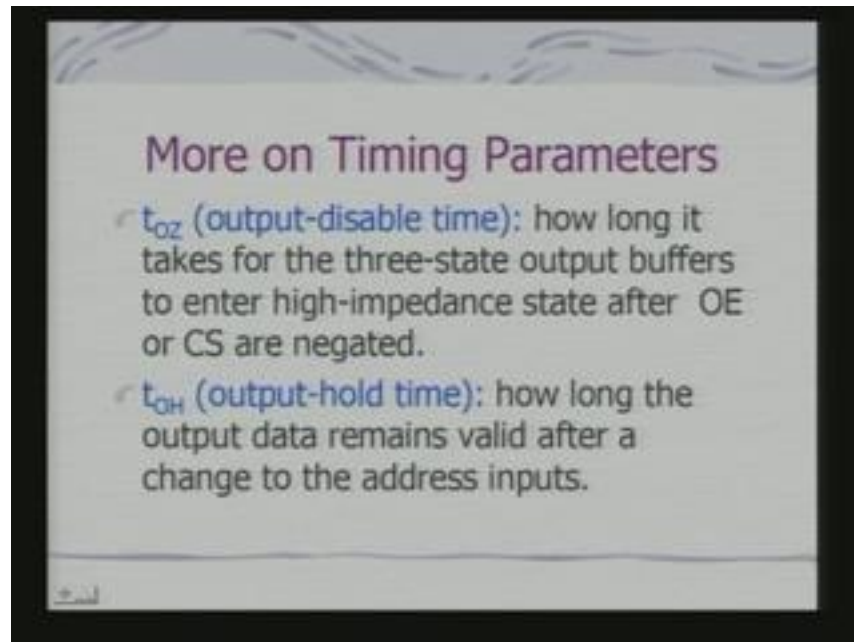
(Refer Slide Time 13:15 Min)



This is the  $t_{OZ}$  which is output disable time- how long it takes for the 3 state output buffers to enter high impedance state after OE or CS are negated and  $t_{OH}$  is output hold time how long the output data remains valid after a change to the address inputs. Now, these parameters that I have discussed in detail are primarily read timing parameters. Now, what is their importance; if I am connecting a processor to a RAM memory block for the purpose of reading and writing obviously, the timings have to be consistent and these parameters determine the speed at which I can really read the data from the RAM or write data on to the RAM.



(Refer Slide Time 14:05 Min)

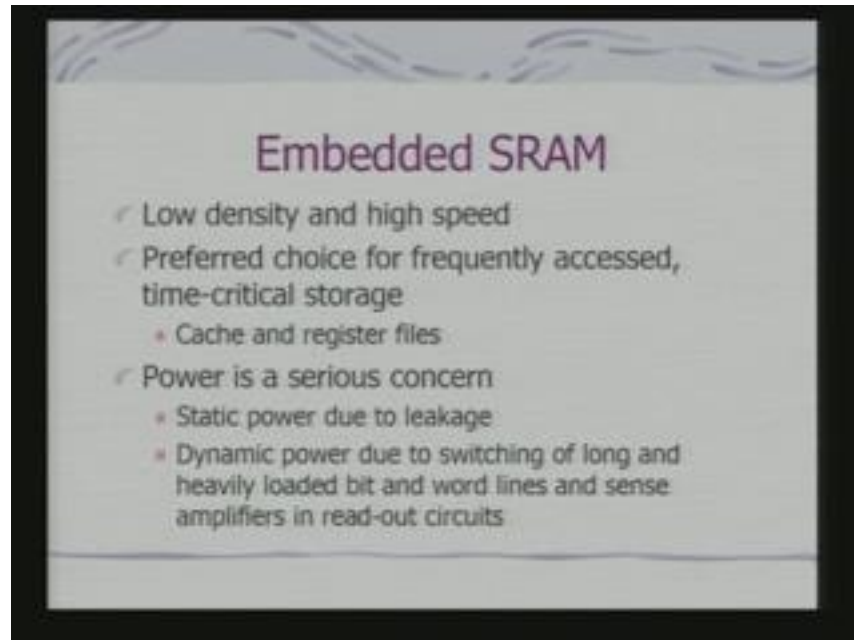


Now, we also have seen that we have got SRAM which are already embedded inside a micro-controller chip and when such an extract gets embedded into a micro-controller chip, basically the designer has taken care of all these timing requirements. See, you have got a compatible memory on chip, externally when you are interfacing, your selection of memory block has to be dependent on these parameters and that will also dictate the speed at which external transactions can take place. Typical feature of an embedded SRAM would be obviously low density because I am using 6 transistors for bit and high speed and this is a preferred choice for frequently accessed and time critical storage like cache memory and register files and in fact you have seen like in processors like PIC. Your entire memory is organized as a registered file and there it should be basically static RAM but wherever you are using a static RAM, power is always a serious concern because in an embedded system we would like to minimize usage of power.

In a static RAM what are the source of power loss or power consumption? One is static power loss due to leakage because you have seen the transistors on the path, okay, there will be a leakage current even when you are not actually accessing that memory because that memory is in an active state, it is storing the data that you have store. So, there will be leakage current and that is what we call a static power loss. Then there will be a power consumption, what we call a dynamic power consumption due to switching of long and

heavily loaded B10 word lines because the B10 word lines are connected from cell to cell obviously I shall need current to drive this B10 word lines to provide meet the correct data. So, there will be switch dynamic power consumption that current will be driven into those lines when I am actually selecting or reading a word from the memory, writing a word into the memory.

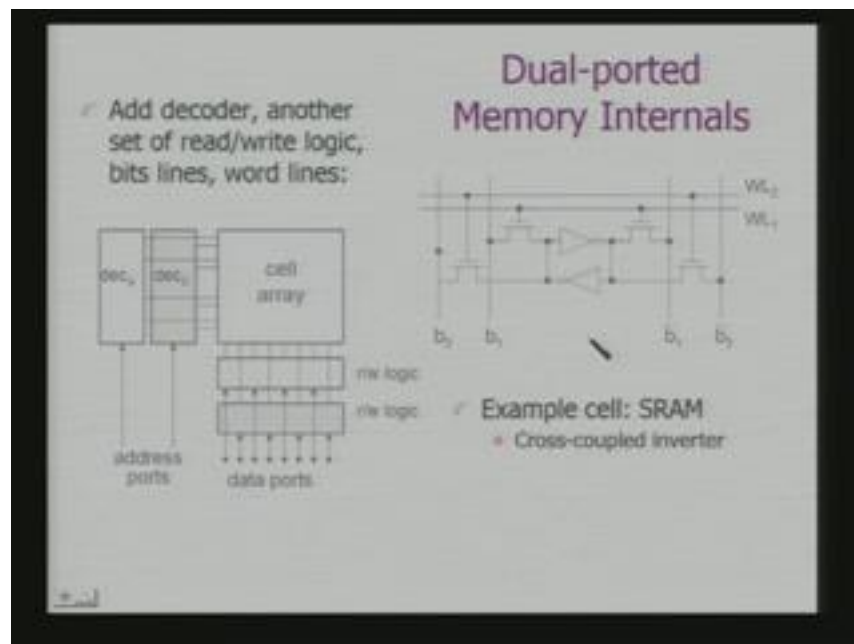
(Refer Slide Time 16:45 Min)



Also the sense amplifiers which are, which comes into play when you are reading out, their speed also depends on the power that they are consuming. So, the 2 components actually account for power consumption by the S frame. So, power is always a serious concern when you are using SRM in an embedded system. So, there are various technologies which we have been developed to minimize the power consumption but not sacrificing the speed. Next thing that we have seen, that in many embedded processes, in DSP processes as well there are multi ported memory. So, obvious motivation in any kind of a pipelined processor which are the processors which today we find to be, find them in many of these SOC's and embedded systems are pipeline processors. If I have one read or write parts cycle, it can limit processor's performance. It also complicates the pipelining because it becomes difficult for different instructions to simultaneously read or register file or on chip memory. So, very common arrangement is for pipeline CPUs to be

provided with 2 read ports and 1 write port associated with the memory. So, you can see that we have port in from cases dual access memory, single access memory. In dual access memory, there are basically 2 read ports and 1 write ports, it is a common arrangement. So, there are simultaneous reads from the memory block. So, effectively what does that mean, if I am talking about a multi ported dual port; effectively it means that I am having additional decoders. The 2 decoders, they are because there can be now address coming from 2 distinct buses because there can be 2 simultaneous read access from a memory block. So, there are 2 distinct decoders and another set of read, write, logic bits lines and word lines. So, you have got this 2 distinct data ports; so this is where the read, write logic will come in and in this case the picture is something like this- you have got the same inverter, okay and now you have got in this case it 2 bit lines, this is a bit and bit bars, okay. So, you can have simultaneous access because I can sense B2 and B2 bar and read the data, similarly B1 and B1 bar and read the data from this cell. So, I got now dual ported read access from SRAM. In fact dual ported memories are very common in SOCs and on chip memories for various DSP processes as well as register, the register bank implementation there also through this kind of dual ported SRAMs.

(Refer Slide Time 20:07 Min)



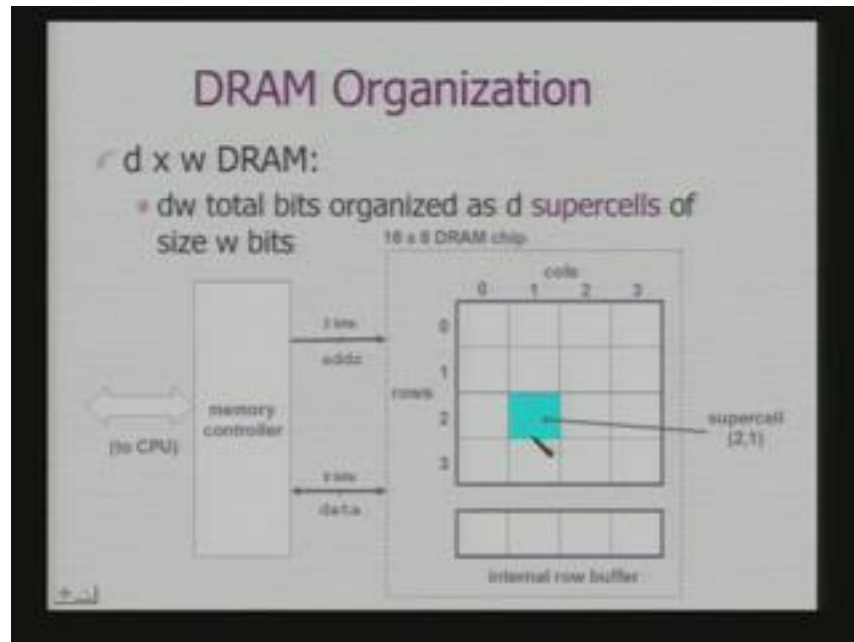
Next, we look at dynamic RAM; so SRAM's exhibit high speed but poor density. So, DRAMs are expected to have high density. Why high density? Because they are actually using a simple **most** capacitance to store data, in fact the charge stored in the **most** capacitance would be indicative of the data that is being stored there. So, simple transistor capacitor spear that is that want we have shown here and this is a basic structure of the cell. So, if you find that this is much less space consuming, okay but the only problem is I need to refresh it at regular interface. Why I need to refresh in a regular intervals, if I am storing charge in the capacitance, the charge will leak out, okay. So, I need to refresh the charge on this capacitance at regular interval. So, it is not at once right and forgotten forever just like SRAM as long as the power is on the data would be retained but this is not true but dynamic RAM because charge can get leaked out. So, it is needed to the refresh at regular intervals.

(Refer Slide Time 21:25 Min)

The slide is titled "Dynamic RAM" in a purple font. It contains a list of three bullet points on the left and a circuit diagram on the right. The bullet points are: "SRAM cells exhibit high speed/poor density", "DRAM: simple transistor/capacitor pairs in high density form", and "Refresh at regular intervals". The circuit diagram shows a "Word Line" at the top, a "Bit Line" in the middle, and a "Sense Amp" at the bottom. A transistor and a capacitor are connected to the Word Line and Bit Line, respectively, and are circled in blue.

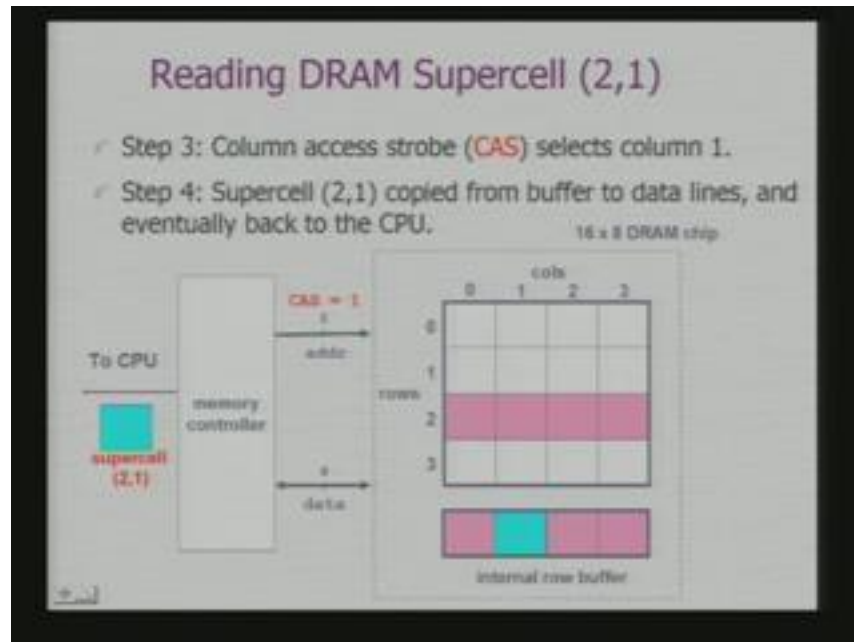
The organization is typically something like this- so you have got, here we are showing a simple example so, what you see, the DW total bits are organized as D super cells of size W bits. So, it is a kind of a row column organization. So, each cell now has got a row address and has got a column address and the CPU generates the address which goes to the memory controller.

(Refer Slide Time 22:15 Min)



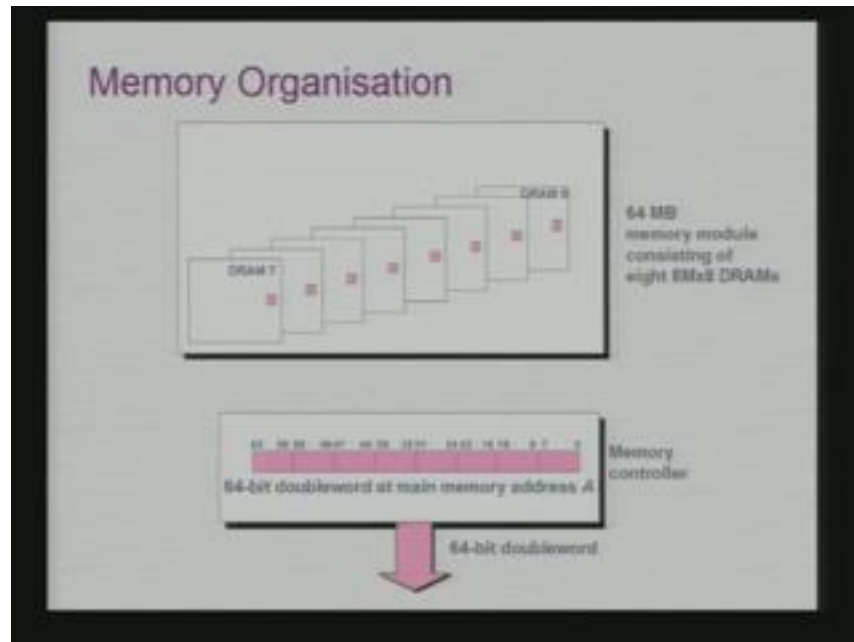
Once it receives the address the memory controller generates, it separately sends the row address and the column address to access the data which is stored in a particular cell. So, let us see how the reading really takes place. First you have to activate what is called row access strobe, okay. So, if I am reading, an example is I am reading the data from the cell 2,1. So, you activate the row access strobe which selects your row 2. So, these gets selected, this is a particular row gets selected on the basis of these signal. Next is, the row 2 is copied from DRAM array to row buffer. So, this will be copied to the row buffer. Now the interesting thing is that the row as a whole gets copied, okay. So, in the buffer you have got data about not only a single cell but multiple cells. So, now I have to select a particular cell using the column address. So, I shall activate the control signal which is column access strobe. So, this column access strobe, okay is activated now and what is to be noted is that in this case this address pass using the multiplex between row address and column address. So, on the basis of this column address you select the internal cell and where do we select that cell. You select that cell in the row buffer and then that copied from buffer to data lines and eventually back to the CPU. So, this gets copied to the data line and then it goes from here to the CPU via memory controller.

(Refer Slide Time 24:25 Min)



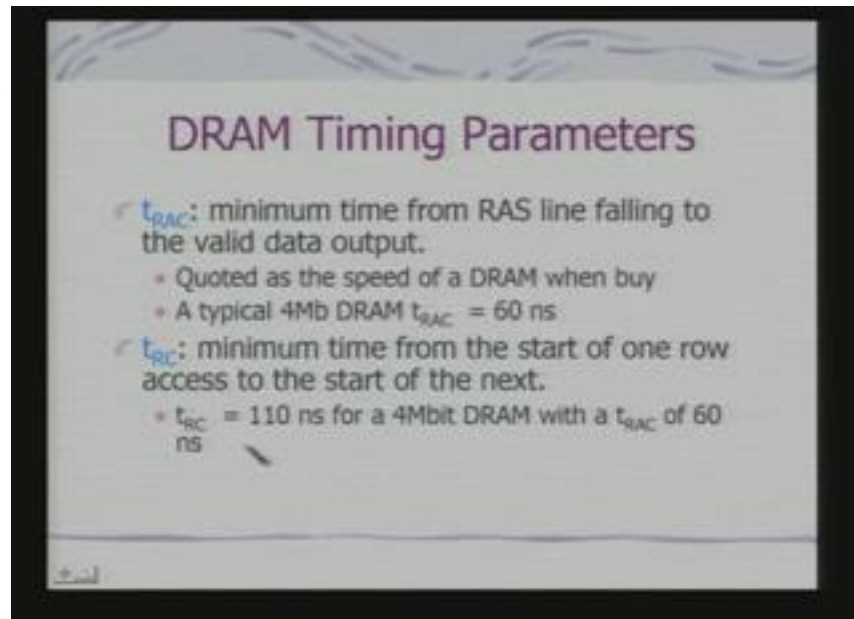
This is the typical read operation for DRAM, similar is your write operation and in fact read operation has got another interesting feature to note, that the refresh operation is normally nothing but dummy read. If I do a dummy read I can actually refresh the cells. How is that memory really organized? So, here I am showing 64 MB memory module consisting of eight- 8 M X 8 DRAM's. So, you generate the address, address is row and column J; so effectively all these cells actually gets selected. Once it gets selected, you provide those bits; you have provided row as well as the column address. So, these cells will provide these bits, okay and all these bits together will constitute the 64 bit or a double word at memory address A and then this will be sent to the CPU. So, what is typically being shown is how multiple DRAM modules can be organized together to provide you at a double word a 64 bit width data word, double word data for the CPU.

(Refer Slide Time 25:40 Min)



Now let us look at DRAM timing parameters. Obviously this timing parameter should be related to rase and cache signals TRAC is the minimum time from RAS line falling to the valid data output, okay. So, if it is, it is quoted as a speed of a DRAM when you buy, this is T that is minimum time from RAS line falling to get the valid data output. A typical 4 MB DRAM- TRAC could be 69 of seconds. TRC is a minimum time from the start of 1 row access to the start of the next. That means when I have accessed 1 row, okay and then I have to access the, another row, there has to be a definite delay or the time period of the cycle and these values typically 110 nanosecond for 4 MB Mbit rather DRAM with a TRAC of 60 nanoseconds.

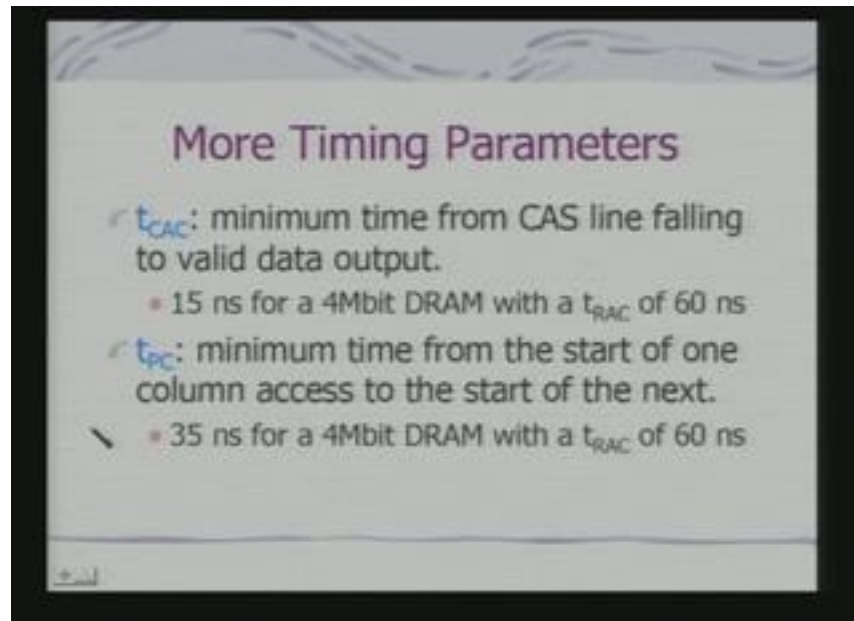
(Refer Slide Time 26:45 Min)



Then, you have got minimum time for the CAS line falling to valid data output, okay. So, falling actually we are, we are telling that is activating; so it is, we are considering all these RAS and CAS as active low. So, it is 50 nanoseconds for 4 MB DRAM with TRAC of 60 nanoseconds; so that is the typically order of the timings and TPC is a minimum time the start of 1 column access to the start of the next, okay.

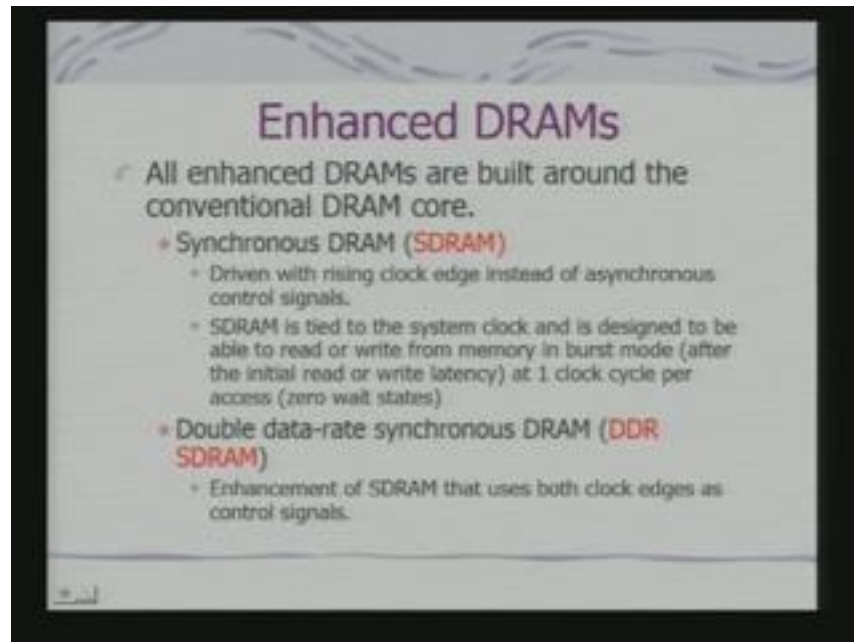


(Refer Slide Time 27:20 Min)



You can see therefore the time taken for a data access for the DRAM will be more because you have to give time for the row access and then the column access. There are a variety of enhanced DRAMs which have been proposed recently and you should understand what is the motivation for this kind of technological development happens. The whole access of DRAM is based on 2 asynchronous signals which are RAS and CAS and they are not really synchronized with a bus clock because once the address is provided, okay to the memory controller these RAS and CAS signals are being generated and then with respect to RAS and CAS, your row address and column address have to be put in to access the data. Now, the variations of these to speedup the DRAM access have been suggested which is synchronous DRAM or SDRAM and in fact in many of the PC's and others today when you buy memories, you buy SDRAM. So, it says the RAM is driven; the control signal is based on the rising clock edge instead of asynchronous control signal that is RAS and CAS. So, SDRAM is tied to system clock or the bus clock and designed to be able to read or write from memory in burst mode after the read or write latency at 1 clock cycle per access.

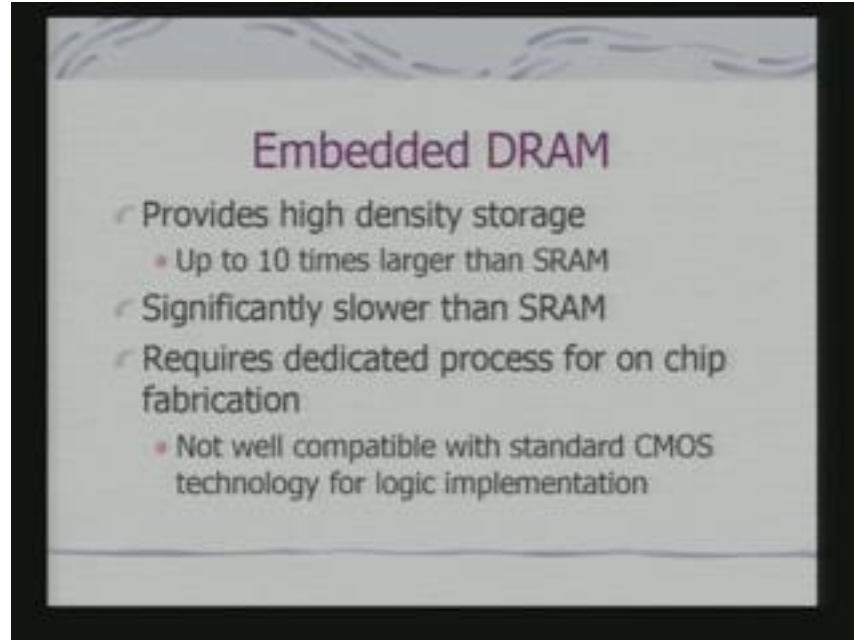
(Refer Slide Time on 29:10 Min)



So, you do not need what is called wait states because if RAS and CAS are asynchronous signals and they are generated by the memory controller; so that time taken to get the data could be more than the bus clock or the clock at which the system or the system clock, the clock at which system operates. So, system would need to wait to get the data from DRAM and that is why the wait states can get inserted, but if now this RAM can be made synchronized with the bus clock or the system clock, then the wait state requirement can be eliminated and that is the basic motivation of designing the synchronous DRAM. An improvement on that is double data rate synchronous DRAM, this enhancement of SDRAM that uses both clock edges as control signals. The key feature that you should note here is that we say that data is being accessed in burst mode after initial latency, okay that means multiple data is obtained; okay so that you are effectively you are accessing one data item for clock cycle, fine. How that can happen? This can happen because if I am able to get multiple bytes that is one side select a row, I am actually selecting multiple cells on that row, I can also get multiple data from that row. So, that is a basic intuition behind designing burst mode access to these DRAMs. So, obviously when we have embedded DRAM going on to SOCs. It is primarily for providing high density storage although it is significantly slower than SRAM and the another difficulty is, it requires dedicated process for one chip fabrication because it is not really

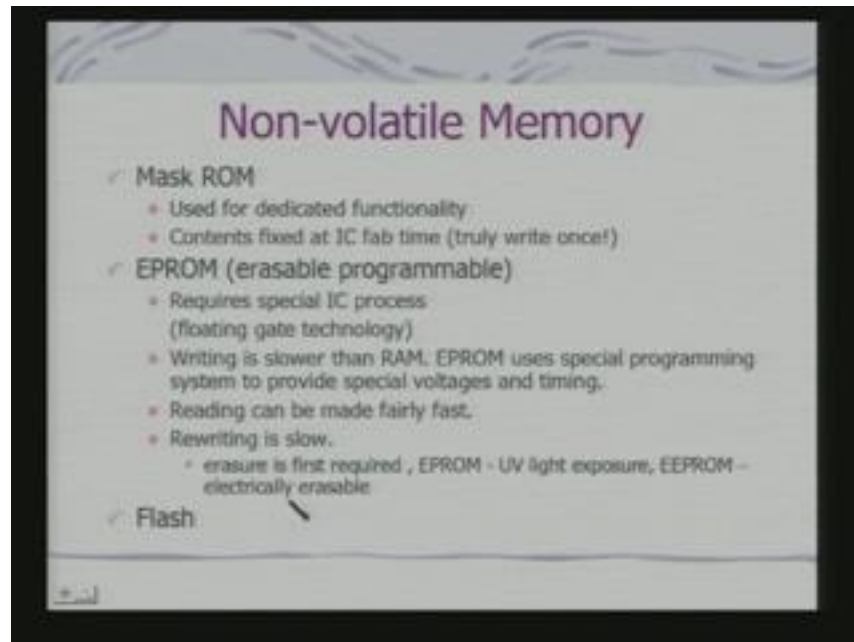
compatible with the standard CMOS technology which goes into fabrication of your logic modules which are your processors and peripheral controllers.

(Refer Slide Time 31:09 Min)



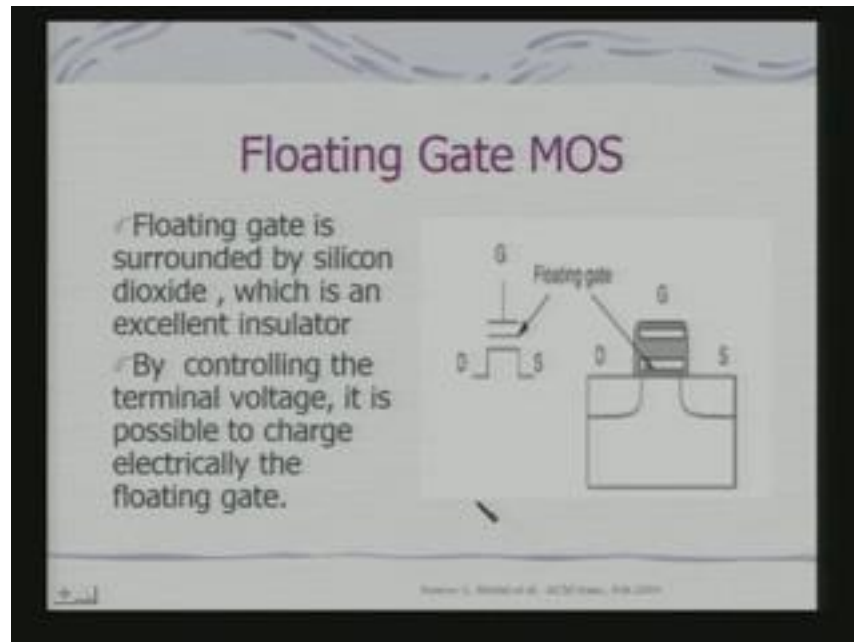
Next, we have non volatile memory, mask ROM are typically used for dedicated functionality and it is once programmable ROM basically, so contents are fixed at the fabrication time itself. EPROM are erasable as well as programmable and they require some special IC process because it is based on what is called floating gate technology. So, this is again a mask technology and floating gate technology; here the writing is slower than RAM and it uses special programming system to provide special voltages and timing. Reading can be made fairly first but rewriting is slow because you have to erase and you typically talk in terms of UV light exposure for eraser for EPROM and electrically eraser erasable modules are called EEPROM.

(Refer Slide Time 32:15 Min)



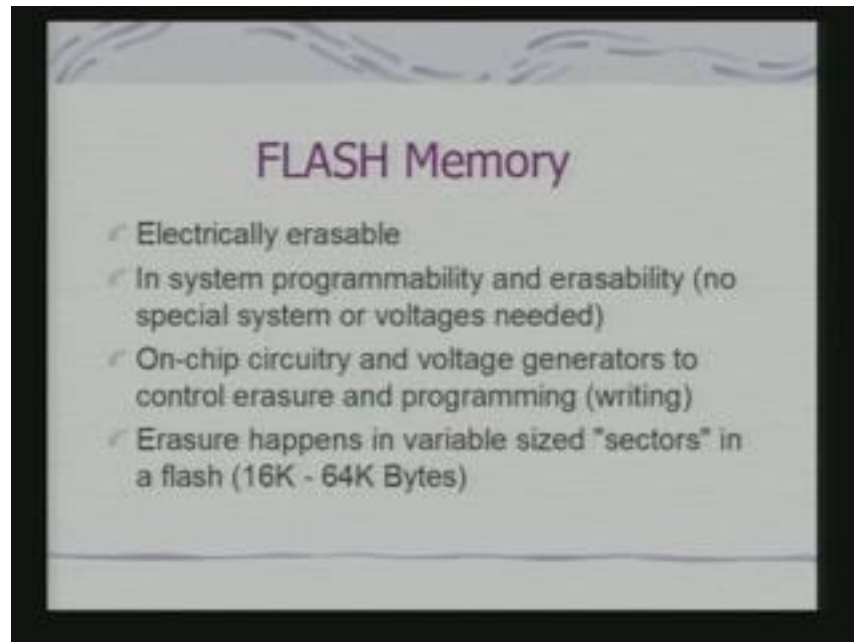
Flash is a variant of this EPROM, so what is the basic technology goes into them. The concept is that of what you called floating gate MOS and it is effectively a single transistor, okay. And just like we say in DRAM, we use a capacitance to store charge as indicative of the data; the same thing is true here as well. All these modern, EPROM, electrically erasable PROM and FLASH memories are built around this technology. There is an additional Floating Gate in these transistors. This Floating Gate is surrounded by silicon dioxide which is an excellent insulator, okay. So if I, if these gate is charged then charge will not strictly flow out only curve. By controlling the terminal voltage, it is possible to charge electrically the Floating Gate and that is why you required special voltages to write on to the chips, okay and obviously there has been enhancement and circuitry going in making than electrically erasable in terms of each bit or each block. So, the basic technology is that of Floating Gate MOS. So what you can observe and realize that here also a single bit is requiring area that of is single transistor. So, the storage density for this kind of non volatile memory will also be more than that of SRAM.

(Refer Slide Time 33:50 Min)



So, electrically erasable PROM are erased using higher the normal voltage because then you can force the charge to flow out and this charge can be rewritten, can be erased by words and not in entire team. What does that mean; not the entire memory area has to be cleared but each word can be cleared. It is in circuit programmable and these as the typical times for EPROM. The flash memory again uses a single transistor per bit. Electrical EPROM implies 2 transistors more than 1 transistor. A FLASH memory provides high density storage with speed marginally less than that of SRAM's, speed is primarily with reference to access of the read time. The write time is significantly higher compared to DRAM. So, they are electrically erasable, but the interesting thing is their erasable in terms of sectors variable sized sectors in a flash. The basic difference between electrically erasable PROM and flash is that in electrically erasable PROM you can erase each word individually but here it is block erasable typically.

(Refer Slide Time 35:10 Min)

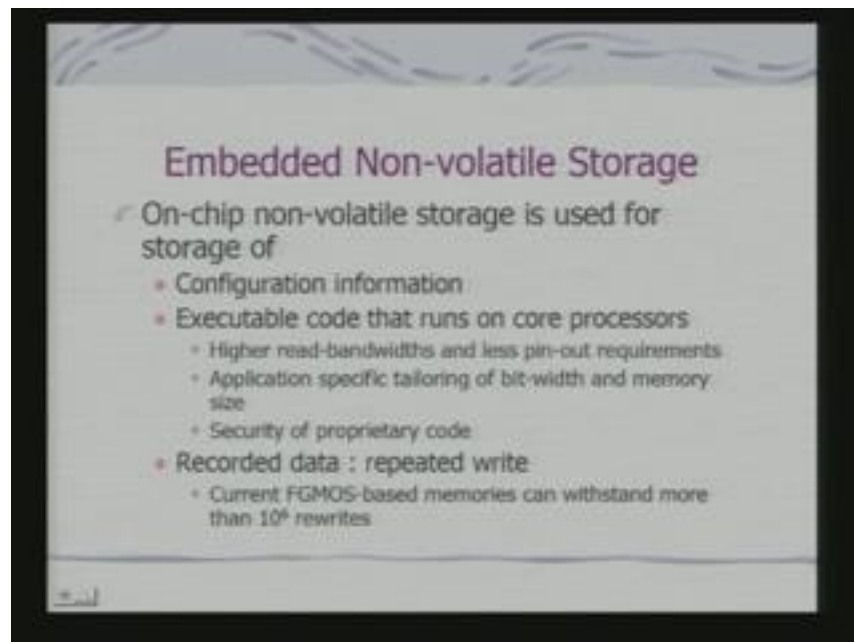


There are two kinds of FLASH which are available which are called NAND flash or NOR flash depending on actually the way the cells are organized. So, you can understand here, this is a typical NAND organization of the cell, okay, this is an individual cell and they are all connected to the common source access point. This is basically the contact point and this is a NAND organization of the FLASH. And this is a NOR organization of the FLASH. So, you can realize the connectivity pattern is search that NAND FLASH could recover smaller size for a fixed amount of storage. So, you have got small chip size. In many cases micro-controllers use NOR FLASH. So, here the area required is typically more and where are these embedded non volatile storage used in embedded system? Obviously they are used for storing configuration information, also the executable code that runs on these processes are in most cases stored on these non volatile memory area.

Now, these gives you can realize the higher read bandwidths and less pin out requirements, why because you do not need an external memory. So, what is being compared here, if you having your program running on an external memory block visa wise the program being stored in a one chip FLASH or any other form of non volatile memory.

So, you can have higher read bandwidths, okay and pin out requirements goes low; you can have applications specific tailoring of bit width and memory size because you do not need to have it compatible with available chips, FLASH chips and their configurations. The other interesting thing which is important is security of proprietary code since it is sitting inside that port it becomes difficult to access and know what the code is all about and then do some kind basically forgery in the sense of taking the code and using it for other purpose. Other thing is a recorded data, typically your voice recorders, your cameras they all use flash which actually enable repeated write. Current this FG MOS base memories can withstand more than  $10^6$  rewrites and even other observation is that one charge they can actually, the charge leakage time can be as large as 10 years. So that is why many of these recorded data's that when you have a speech recorder or a camera you actually have these flash cards to store the data or the flash memory to store the recorded data.

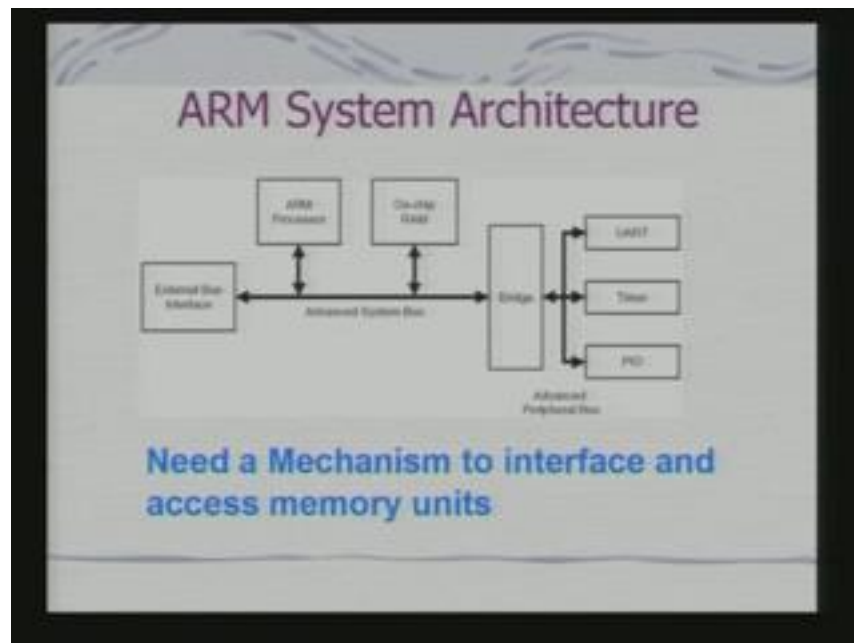
(Refer Slide Time 38:00 Min)



Now, we shall see that once we have got these types of memory, we shall now see an example of interfacing memories, okay with the micro-controllers that we had already studied, in particular we shall look at OR. So, if you look at ARM system architecture, go back to the ARM system architecture obviously I have got on-chip RAM, okay and we

know how to use, we need to know how to use this on-chip RAM but we can also have external RAM to be connected to the system. The whole idea here is that when we are interfacing external memory units, we should know how do use the signals which have been provided by the processor core. Now, this is important if when we are building up a system in terms of discrete components. This is important even when we are building an SOC because when am building on SOC I have got I have got this ARM core as a specification and I know what kind of signals have been provided by the core.

(Refer Slide Time 39:33 Min)



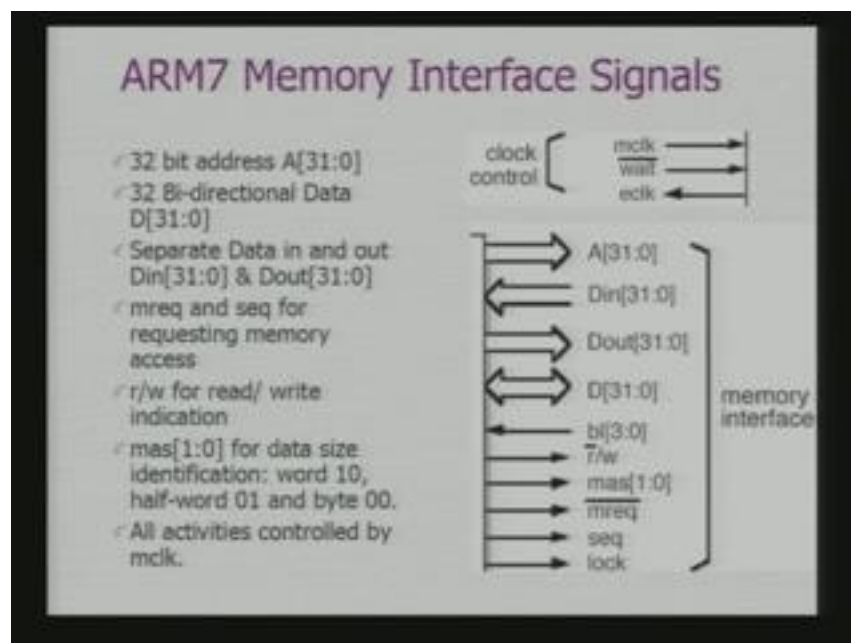
So, if I am developing a memory module to be used in my SOC, that memory module has to be compatible with the signals that can be provided by the processor core. So, we shall look at ARM 7 memory interface signals. In fact we had already looked at them; we are just reviewing it again for the sake of completeness. It has got at 32 bit address bus it has got 32 bit bidirectional data bus as well as separate data in and data out bus; so that been depending on your requirements you can connect your memory modules to these buses. Now, very obvious thing would be that if you were just putting in a ROM where you do not need to write, okay an non volatile storage where you do not need to write, you may connect it directly onto the data in bus and if you are connecting a RAM that would be typically connected to may be connected typically to a bidirectional bus. Although if, if it



supports separately data in and data out pass, if the external memory supports explicitly data in and data out lines then you can connect it here as well but many of the memory modules have got a single bi directional data interfaces.

This mreq and seq, these are two signals, okay this is an active low signal, m requires this actually indicates whether your ARM core is requesting memory access or it is requesting sequential memory access. Now, these points we shall see that how to use this signal for having an efficient access when we are interfacing with external memory. Read, write or for read write indication, this one we will look at the size if you remember we started the basic model where we had size as the signal. So, these signal this is a two bit signal because it indicates the different sizes so these indicates the size of the data which used to be transferred over the bus and these all activities at to be synchronize with regard to the mclk that is the system clock. Now, here you can find out that the clock control has got associated with it s signal called wait, but this an active low signal. In fact your internal clock for the processor core is actually combination of the 2 and how and when to use this wait signal.

(Refer Slide Time 42:45 Min)

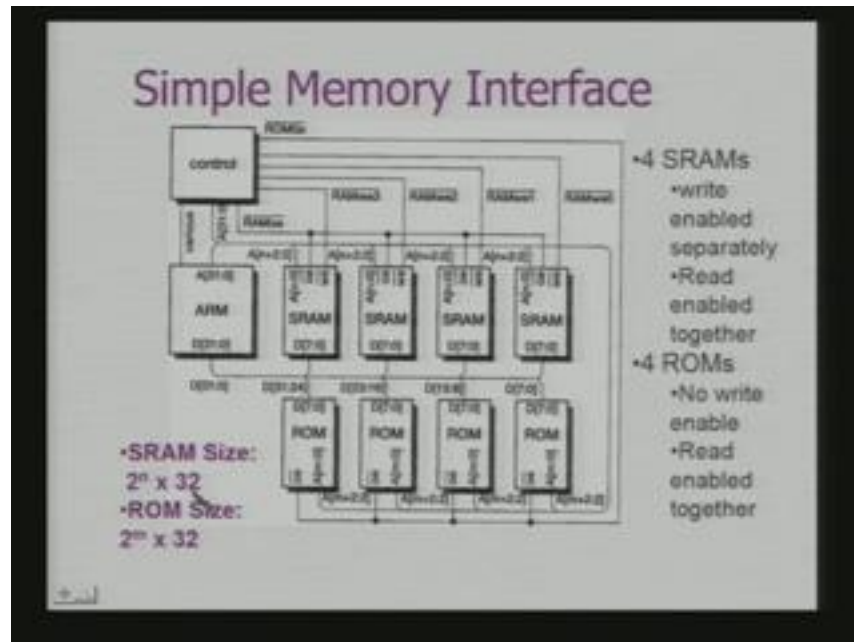


If you remember I said that in many cases, my memory may be slow; when my memory is slow, for a memory access the processor has to wait. So, the memory module has to tell

the processor that you please wait till I provide the data, okay. In that case this wait signal becomes important. So, that is why I am telling that your internal clocking or internal operation is controlled through a combination of mclk and wait signal. Let us see how we can connect a simple SRAM and a ROM module. This is at typical ARM core and this is a control logic that I can built, I can connect this SRAMs, these are all 8 bit SRAMs, okay and these are all 8 bit ROMs. Now, what we have seen here that write enable this SRAMs are write enabled separately. So, that means typically here I have got individual write enabled signals. The individual write enabled signals are not coming from the ARM core that is to be generated by the interfacing logic which is here, okay and here what you are finding, this address is going to all of them and what I have shown here is that your address is provided to all these SRAMs and they are read enables. The read enables are all enabled together, okay. So, what we are telling is the read enabled together but write enabled separately. Similarly, in case of these ROM, no write enable signal is required because I do not need to write to them and they should be read enabled together.

Now, here what you will find is that we have shown typically if you see the address buses, the addresses the way it has gone in that this is a 32 bit address, okay and this 32 bit address is fed to the controller as well as to the, these RAMs because these address have to be decoded by these RAMs to select the individual rows. The decoding logic is part of these SRAM module, okay and this is being fed here and the reason why you say that the last 2 bits are not being used because if they are all located, if these are all 32 bit word data and if there are byte addresses and they are to be located at 32 bit boundary, then last 2 bits actually becomes redundant, okay and this is a common, this data bus to which port RAMs are connected, okay. So, effectively it is in terms of 32 bits. Now, the question comes- how do design this control? When I am designing this control what I have to do, I have to use various control signals which is ARM code is producing, okay and that control signals have to be used.

(Refer Slide Time 45:52 Min)

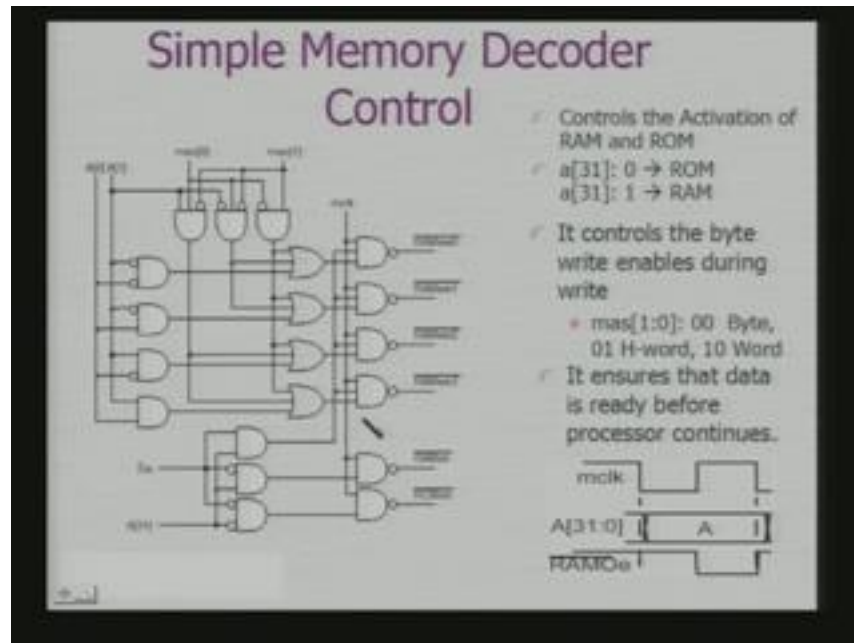


So, this is the simple example of a control signal. Now this control can be implemented on an FPGA or may be implemented in terms of a discrete logic if it is really going into um SOC these becomes part of the SOC. So, the interesting thing to note is, I control activation of RAM and ROM. So, a 31 bit is typically being used for this purpose. So, if you see here, a 31 bit is being used for selecting for RAM or ROM obviously because they are located at 2 different addresses. They have to be located at 2 distinct partitions. Now, the other interesting thing is that these, third one what we have got is, this mas 1:00 byte 01 word huff word 1 0 is word.

Now, these signals are being fed here and the logic is with regard to the different RAM write signals that means depending on the size of access that means I may like to write a byte at a word location, okay. So, if I am writing a byte then I am activating a particular RAM, if I am writing 16 bit I shall activate 2 of them. So, that is why the size signal becomes important and this is coordinated with NCLK signal because what we say that all these things have to be valid, okay RAM enabled when the address is valid and this signal gets validated with respect your basic clock which is driving the system is. Is this clear? So, now these addresses are these address is if you see a0 and a1 which were not being fed to the individual memory blocks. They are actually being used for selection of which signal to enable, okay that is depending on the size of the transaction and

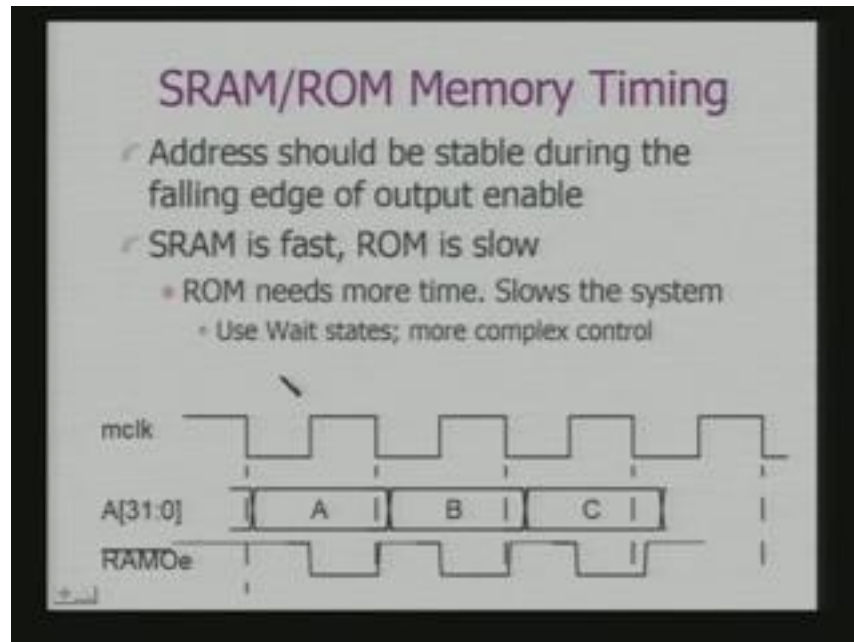
depending on these address, 1 or more of these RAM write enable signals will be activated, okay and this is away by which your size information has to be made use of because you need to know for a bus transaction what is the size of the data transfer that is taking place.

(Refer Slide Time 48:40 Min)



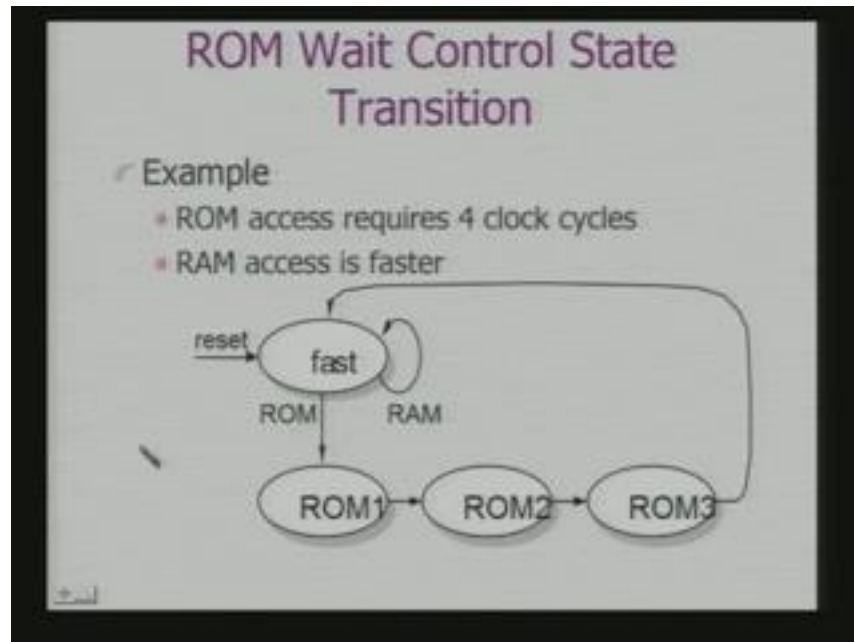
So, address should be stable during the falling edge of output enable. The SRAM is fast but ROM is slow and ROM needs more time, it may, it slows the system. So, what we are trying to say is that the access time for the ROM is slow. Now, that provision we have not shown that how to take care of the slow access time of the ROM. Now, the whole things need to be modified to take care of the slow access time of ROM. So, obvious thing is we use wait states that means when you are using a ROM access you have to use the wait signal so that the system can wait.

(Refer Slide Time 49:30 Min)



Now, so let us take an example; so where ROM access requires 4 clock cycles and the RAM access is faster. Now, obviously in this case, the bus transaction, the control logic for the bus transaction has to be captured by this FSM. So, what we say that if I am starting, so this FSM goes into where, this FSM is actually will go in to implementation of the control logic.

(Refer Slide Time 50:39 Min)



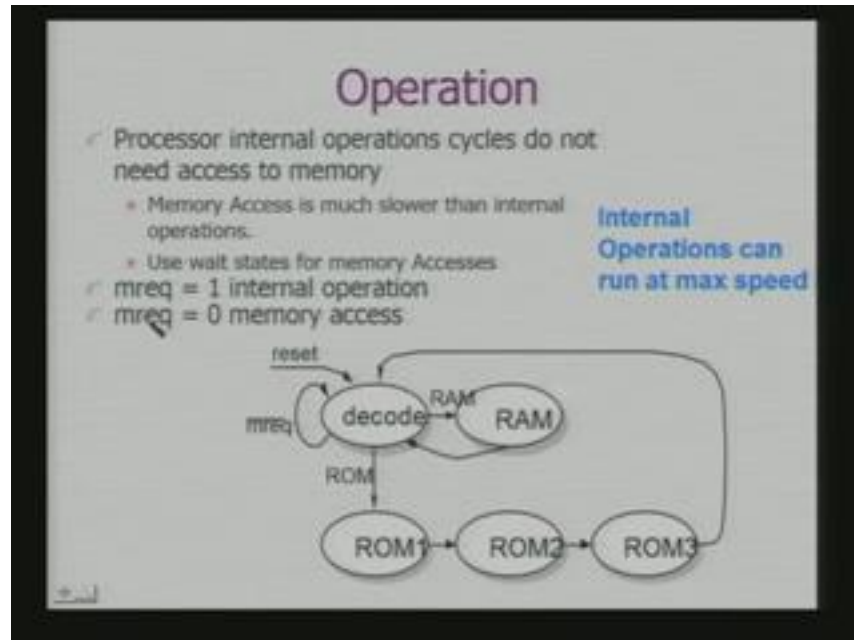
So, if it is a fast, so if it is a RAM, so I shall be always in the fast mode; if it is not so, if it is on the ROM, I have to take care of 4 states. So, my wait signal should be low, okay if it is an active low for 4 states, so that the system should wait, till it gets the data from the ROM. So, this is basically my control logic defining FSM. But now let us look at this timing diagram. So, effectively what we are telling that these wait goes low here, okay and this ROM enable has to be active for this time period because these are the 4 cycles which is required for the access. So, this is now the timing diagram that would be there when I am using a ROM which is requiring 4 clock cycles for an access. But how does the processor operates, will we really need the processor to wait because my processor is actually a pipelined processor because there would be other instructions which can be executed and when I am looking for this data from the ROM because ROM can actually have got the instructions, the programs.

So, while one instruction is fetched, other instructions are getting executed. So, processor internal operation cycles do not need access to memory and in that case the processor performance is not going to suffer, okay that is something which is important; only wait states will get inserted when the processor explicitly requesting the memory access.

So, that is the basic significance of this m request signal which is coming out of the processor, okay. So, m request 0 clearly indicates the memory access and m request 1 is

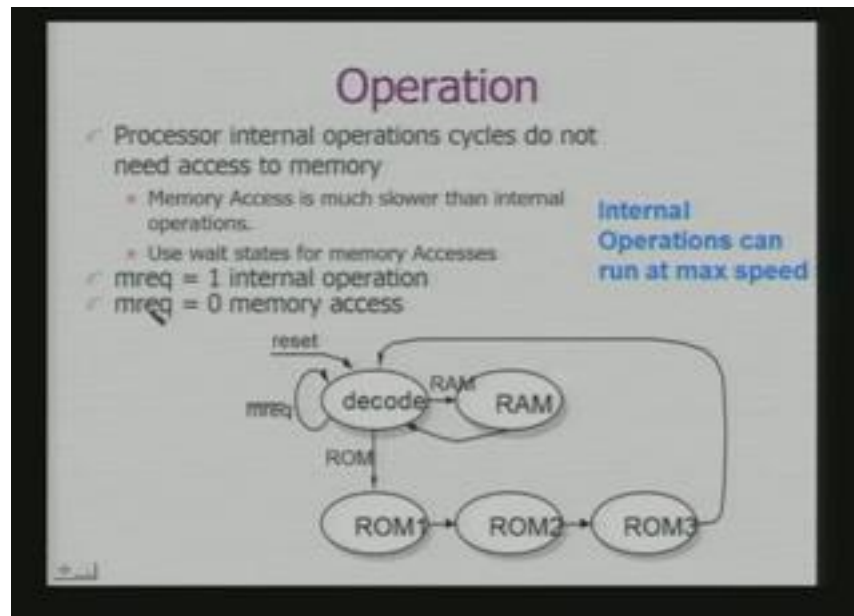
the internal operation. So, when it is m request 0, okay effectively what it means, I go to the RAM decode, I go to the RAM and come back and when I am doing the access here it is a ROM access right and corresponding to that there will be the wait state which should be generated. So, actually these m request signal indicates that at what point the wait should be really made active, okay when we are doing a ROM access.

(Refer Slide Time 52:45 Min)



Then let us look at this DRAM access. The question is what do I mean by the internal operation. Internal operation means it is, when it is executing instructions, say ARM it requires registers which are involved in the operation. So, it does not require a memory access so that operation can continue without the memory access. So, it is not that ROM access would make the processor always suffering for 4 wait states. So, it is indicating through memory required that it is now doing going into a memory access cycle so that memory access part of it can wait because it is accessing the ROM.

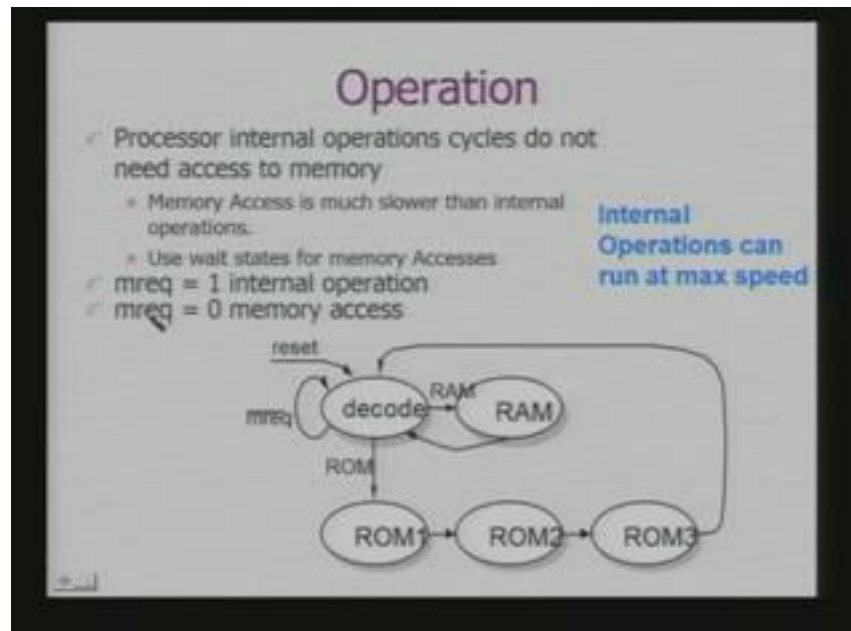
So, let us look at the DRAM and the question of interfacing DRAM with R. We are just recapping the DRAM organization, you have first row address. The first row address is presented and latched by ras signal. Next column address is presented and latched by cas signal. This is what we have already seen. Now, how to do that with ARM? Does it have any spatial signal to facilitate DRAM acce(Refer Slide Time 54:00 Min)



Now, how can I make DRAM access faster? Okay. Accessing data in the same row using cas only access is 2 to 3 times faster because you are not paying for the row access time, okay. So, cas only access does not activate the- what we say the complete cell matrix. If next accesses is within the same row and new column address may be presented just by applying a cas only access and these exploits the fact that most processor addresses are sequential.

(Refer Slide Time 54:50 Min)



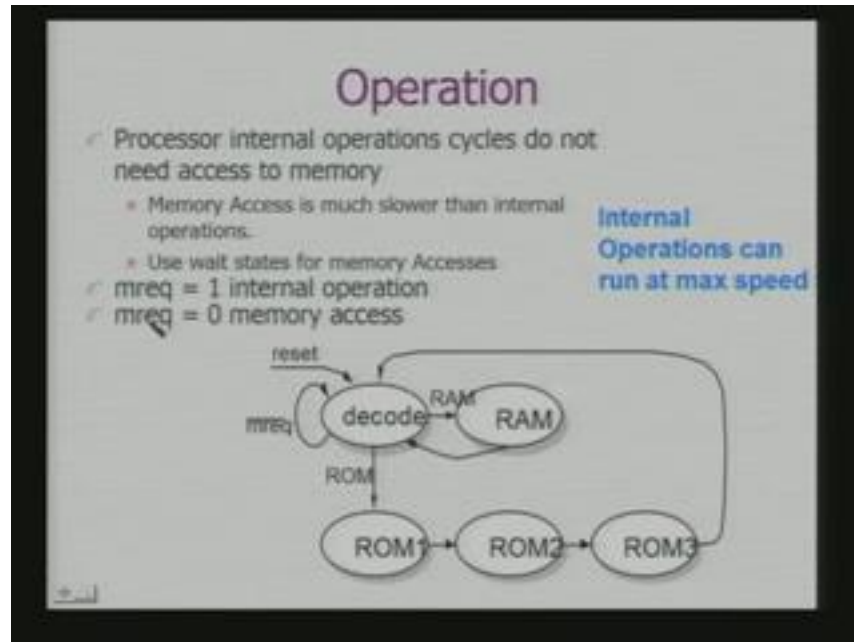


If I am not really executing a branch, okay I shall be accessing addresses in a sequential fashion. If I am accessing addresses in a sequential fashion, then I can have a much faster access from DRAM, okay and particularly if I have my DRAM for storing the data. If I am accessing say data blocks like an array okay in a particular fashion, in a discipline fashion, then I shall be accessing them sequentially. So, what we shall do, that if we had a way of knowing that the next address is sequential with respect to the current address then we would only assert cas and make DRAM access fast.

So, I need to know this next access is a sequential access and so I need to deduct early in the memory access cycle, the next address is in the same row, okay and for that the sequence signal seq signal which comes out of the ARM is an ideal source of that information. So, sequential addresses are flagged by seq signal which comes out of the ARM, fine. So, the external memory device checks the previous address and row boundaries to issue cas only or ras cas combination because all sequential addresses cannot be satisfied by simply only because I may go from one row to another row. So, memory device controller will decide whether to generate only cas or to generate both ras and cas. So, effectively this finite state machine that which defines a controller for bus transaction would look something like this. So, depending on whether it is sequential or not; so I shall, this is the sequential when I am doing here. So, effectively I know that in

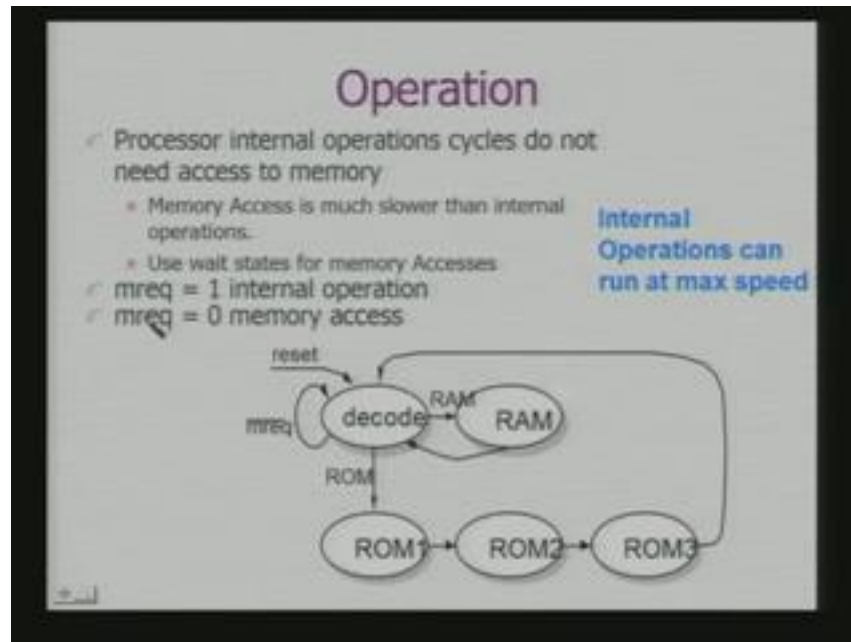
the state I shall activate what, the cas only state and or ras cas depending on the condition. Otherwise when it is a ROM access, I have to go into a wait state and particularly when there is a memory request.

(Refer Slide Time 57:10 Min)



So, the timing diagram would look now something like this. So, in fact it is a pipelined memory access so, if I have a pipelined memory access, typically address is presented half cycle earlier. So, what we have got here is this is your clock- the system clock and these are the addresses which have been provided and here you find that the address is coming earlier, okay. Earlier so that you can know that the next is the sequential address then the sequential, this one is activated. This wait state comes in because there will be a wait when it is a ras cas access, okay, wait for a ras cas access and wait is asserted by the external logic. So, when it knows that there is a sequential access, the wait need not be low. You have the ras, the ras remains valid all through and what you are having you are asserting cas with different column addresses. You are asserting cas with different column addresses and so you are getting the data from consecutive locations and this is precise the reason why the ARM processor core provided seq as an output signal, but you can understand that the way to use this signal for actual interfacing, you need to implement the interfacing logic.

(Refer Slide Time 58:47 Min)



So, this comes to the end, this brings us to the end of today's lecture. We have learnt about different types of memory and their characteristics. We have seen how external memory can be interfaced in ARM7 based system. We shall next study memory management and use of memory, because use of memory is a very critical issue in the context of embedded system because I have to have limited memory and I need to know how to use that limited memory effectively, in the future classes.