Multi-core Computer Architecture – Storage and Interconnects

Dr. John Jose
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

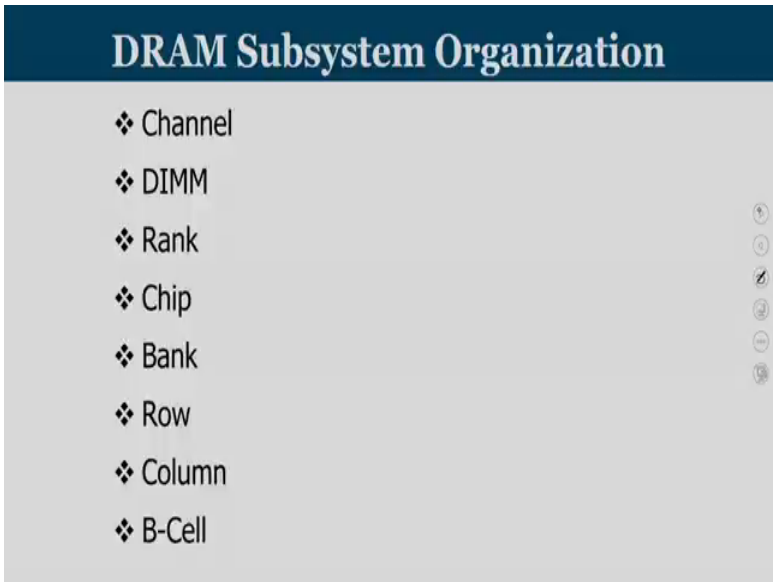
Lecture – 10
DRAM Controllers & Address Mapping

Welcome to the next lecture of our course Multi-core Computer Architecture-Storage and Interconnects. Our discussion on the last lecture was about the basic organization of DRAM module. And, we have seen, what is the role of the DRAM and how it is internally organized with various hierarchy right from channels to DIMMs all the way up to rows and columns.

Today, we will see one of the important aspect with respect to DRAM design, that is about address mapping; that means, once you have an address how will you find that whether this address is under which channel or it belongs to which rank or bank etcetera. And, the other component very crucial component that drives the entire DRAM is the DRAM controller; it is the one that generates appropriate control signals for the DRAM to work.

So, once this controller receives the input from the processor side or from the last level cache controller. It has to generate appropriate control signals such that DRAM will work accordingly and retrieve the data. So, today our focus will be on DRAM controllers and address mapping.

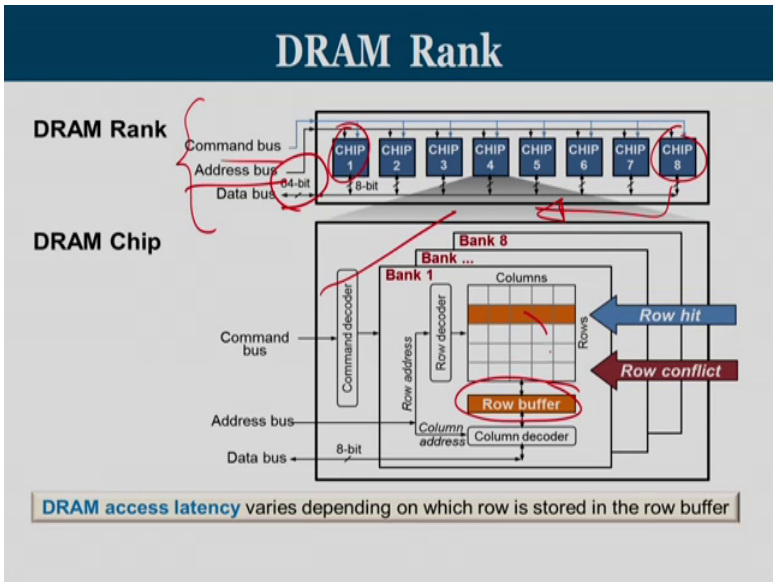
(Refer Slide Time: 01:50)



We have already seen that the DRAM sub system consists of channels, which are basically data busses. So, if you multiply channels you have more number of connecting points from the DRAM to the processor. And, then we have multiple DIMMs are there each channel can consist of multiple DIMMs. And, DIMMs are further subdivided into 1 or 2 ranks and ranks will be consisting of mini chips. Chips consists of multiple banks that is arranged in a 3 D fashion and each bank we have it is own row buffer and bank is organized as rows and columns.

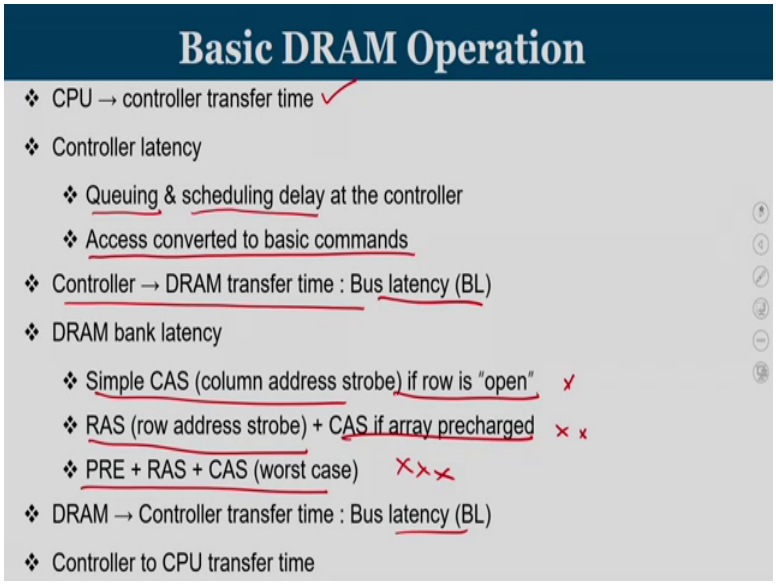
So, we have seen that once you give a row address, the contents of the entire row is getting transferred to the row buffer and then you give the column address and the mentioned what kind of operation weather it is read operation or a write operation, that you are going to carry out on this DRAM.

(Refer Slide Time: 02:50)



This diagram is also familiar we can seen that a DRAM rank is a unit where in common address and commands are being given and different chips are going to return with a different subunits of a given data. So, all together if it is 64 bit data bus 8 bit which is given and you have seen the bank structure it is 3 D structure that is going to organize your data stack wise and you have a row buffer which is there in each bank. So, the condense of a row you are transferred to this row buffer.

(Refer Slide Time: 03:27)



Now, we will see about basic DRAM operation, when they DRAM is required? We know that in order to carry out the instruction execution with the help of an instruction pipeline, during the fetch stage itself we will contact the cache memory and multiple levels of caches are there, if the you required instruction or data is not available even in the last level cache then we require the help of a DRAM.

So, CPU essentially the processor chip which houses of the processor as well as the caches has to communicate and that communication is done to the DRAM controller. So, there is a transfer time that is involved, from the CPU chip on to the controller in the DRAM control of transfer time. And, the controller itself has it is own latency there are a some kind of an operation, that happens in the controller and that has to be carried out. Now, what are the important components or important operation that we do many request will come to a controller because, we are now already multicore environment.

So, you can have request coming from multiple tiles or multiple processors. And, all these things is going to queue up inside the DRAM controller. And, see how the queuing delay and then you have to pick one among them that is called the scheduling delay that happens at the controller and whatever command you have to generate.

So, once you identify a particular request is going to be service from among many request that comes to DRAM controller, we have to queue it up and from the queue you have to pick 1 that is called scheduling operation. Once scheduling is done let us say request number r 1 is to be serviced first, then these request r 1 has to be converted into a couple of basic commands.

Once the commands are there then from controller you are passing to the DRAM. So, DRAM is outside the controller in all modern day processors the controller is either on the chip set that we can see in the north bridge chip set or the controller can be on the processor itself. So, once the controller is communicated controller fix up what is the request that is to be serviced, appropriate commands are generate at these commands will flow from controller on to the DRAM and that is propagated to the bus that is what is called the bus latency that is going to come.

So, now the request have reached the DRAM unit the DIMM and there you have to spread the addresses column address and row address. So, there we have different cases, if, the row is already opened then only a column address strobe is required; that means,

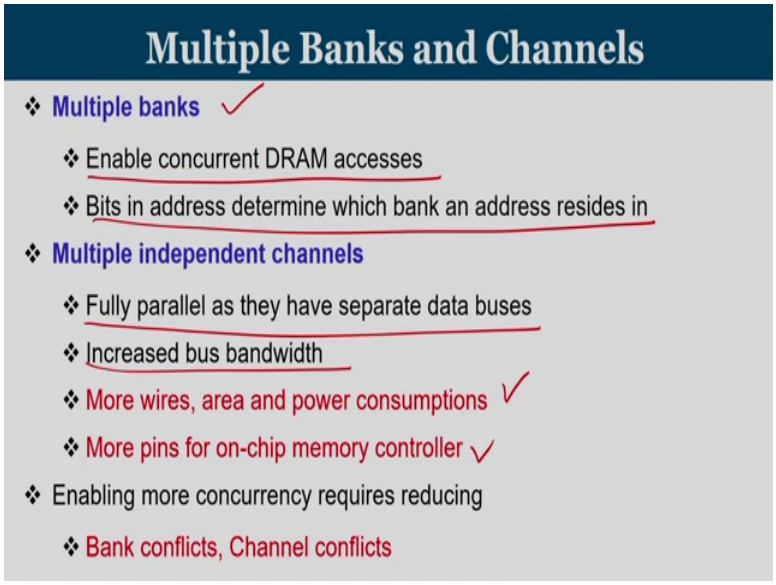
already in the row buffer, we have the condense of that particular row which we are talking about is available. We call it as an opened row scenario; if it is an opened row scenario then we need to give only the column address.

Now, if it is already pre chart; that means, already it is a closed to row then you have to give a row address strobe followed by a column address strobe. So, RAS and CAS has to be given one after another with appropriate timing gap between them. The other one is called the row conflicts some other row is already opened. So, we have to close that row that is being done using a precharge commands so, there is a precharge command that goes, after the precharge command we have to give the row address strobe and column address strobe.

The 1st one will take very less amount of time, the 2nd one will take little bit more amount of time because there is a row address strobe that is involved and the 3rd one it involves a prerecharging operation followed by your row address strobe and a column address strobe. So, this DRAM bank latency is involves it is a varying number, it depends on whether it is an opened row, whether it is a closed row, or whether it is a row conflict.

Once, the DRAM has returned the data now, the data has to go back to the controller and that involves the bus latency. And, once data reaches controller the controller is going to transfer it to CPU or to the last level cache.

(Refer Slide Time: 07:34)



Now, we will see when can you paralyze things, when you get multiple request that is going to come to you or essentially to the DRAM, how are you going to paralyze things. So, one method is using multiple banks, we have already seen that every rank is divided into multiple banks that is in a 3 dimensional layered structure. Now, each of these bank has it is own row buffer. So, when you have multiple banks it enable concurrent DRAM accesses bits in the address determine which bank it resides.

So, if you have data that is available in multiple banks, you can keep already data ready in the row and give the column address strobe. In such a way with appropriate time in delay such that bank one data is transferred, by the time data transfer is over give the corresponding column address for the next bank such that, the data from adjacent bank so, different banks can be sent one after another.

We have to understand that when you have multiple banks we cannot work them parallely, if you have only one data bus that is connecting. So, if multiple banks are connected to the same data bus give the control signals in such a way that, once the data is transferring to the data bus, the data of the next bank should be kept ready. Such that whenever you look into data bus always it is carries data, they may not be given by the same bank there may be given by varying banks, so different bank numbers.

And, we know that for transferring of data you require some row address to be given, so, that the data will reach the row buffer followed by the column address to be given. So, this row address and the column address has to be given accordingly such that data was is never idle. But, if you are going to do whole all the operation in a single bank itself, then after one operation let us say the next address that you want, if it is in a different row then you have to write it back, but precharge it, then give you a new row address followed by the column address.

So, we have bank level paralysm will essentially help you, if you want certain data to have to reach the processor in adjacent clock cycles, it is always better to put them into appropriate banks. So, they should not be put in the same bank.

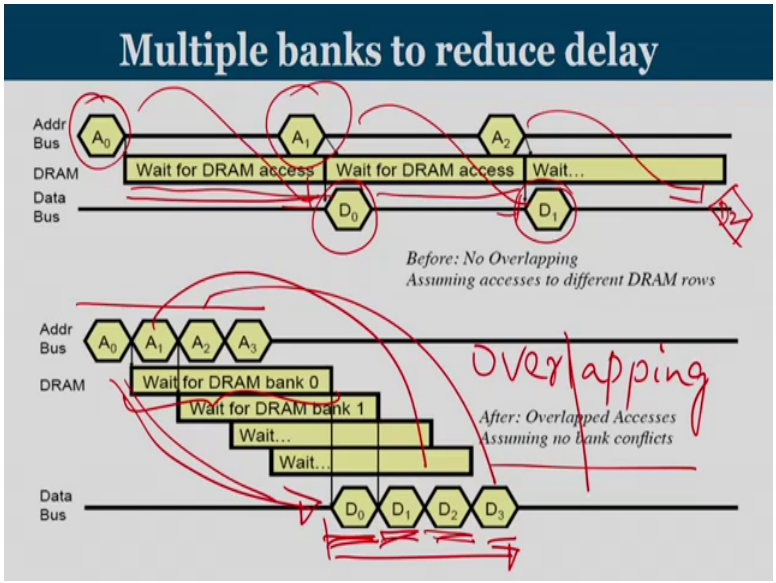
So, in this way the addressing mechanism or the address bits has to be given in such a way such that they are into adjacent banks. Similarly, another set of paralysm happens when you have multiple independent channels, like already mentioned channel is the communication point through which a DRAM communicates with the controller. So,

when you have multiple channels it is essentially you have multiple set of buses. So, this provides fully parallel as they have separate data buses, you should increase bandwidth and you have to essentially use more wires and more area issue are there and more over more pin issues are also there.

Now, when you deal with this band conflict than channel conflict? So, what are essentially conflicts? 2 addresses are coming let us see these 2 addresses belong to the same channel. So, even though you have a multi-channel DRAM system, if the address if the incoming address to the DRAM controller is in a pattern such that both address belong to the same channel, then they can be transferred to DRAM or DRAM can access this location only one after another, because they belong to same channel.

Likewise, if the addresses are in such a way that if these 2 addresses are mapped to different channels, then DRAM could fetch these the content of this addresses parallelly. So, the problem of 2 address mapping into the same channel is known as channel conflict and 2 address mapping into the same bank is known as bank conflict.

(Refer Slide Time: 11:26)



Now, we have multiple banks to reduce the delay, how this is possible? So, consider this case where the timing diagram is given, let us say I am going to give an address these address will reach the DRAM. DRAM has to perform some row address or the other thing at the end of this operation it is go into give you the data.

Let us say you are going to give another address, it takes lot of time because the same thing you have to precharge or row address strobe or column address strobe. And, then that are D 1, D 1 corresponds to data of A 1 and D 0 corresponds to data of A 0. Similarly, you have an A 2 which will give you the data D 2. We can see that after the address is given it has some amount of time delay in returning the data.

This is basically the control and latency or the queuing delay that is associated the scheduling delay, the bus transfer time and the time to give you the raw address strobe, column address strobe and the pre charge signals. Let us assume that the address A 1 A 2 A 3 or A 0 A 1 A 2 this case let us see all are mapped to different banks. So, be give the addresses one after another A 1 A 2 A 3 whatever and each address will have it is own time delay.

So, it will take some time so, that you will get D 0, but since they are all belonging to different banks by the time D 0 is getting transferred, the next bank can be ready with the corresponding data. So, you get an overlapped access that is the delay of each of this is overlapped and if there is no bank conflict, if there is no bank conflict means all D's should be mapped into different banks.

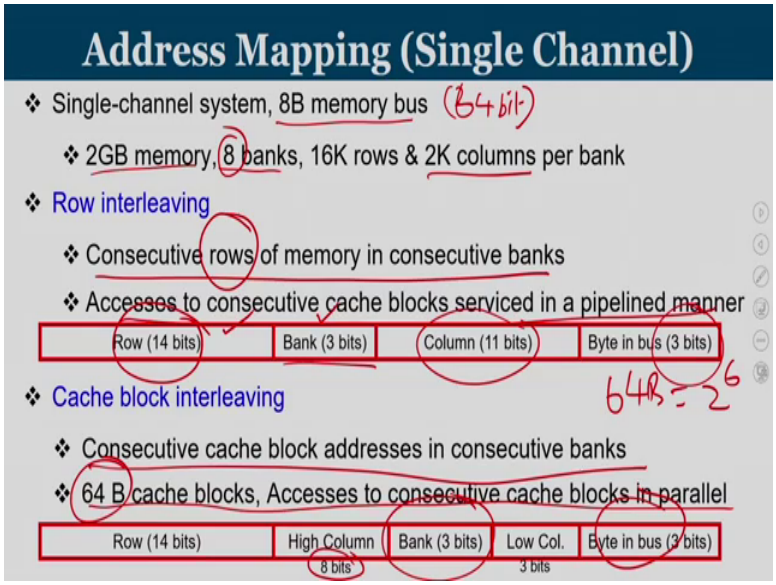
So, once you start data transfer, then without any delay all other data transfers are happening. So, essentially what we have learned from this, for each of the address once the address is given, this address involved processing at different levels. At, first stage in the DRAM controller there is a queuing and scheduling delay and once scheduling is done, once your address is being picked for service. Then it has to be converted to appropriate commands and this commands has to flow from the DRAM controller to reach the DRAM system or basically the DRAM unit to the bus so, that it involves a bus latency.

Now, once it reaches the corresponding DRAM unit depending on whether it is an opened row or a closed row or a row conflict appropriate signals like precharge, row address strobe, or column address strobe has to get be generated. Once you get this then the data is ready again transfer through the bus reach DRAM controller. So, it takes some amount of time after the address is given, it takes some amount of time for this address to be processed and the appropriate data is been retrieved back.

And, again if you give the next address after that, again the same amount of delay is there so, for every address given there is a delay. The question is if the address are mapped to different banks can I overlap the delay, that is exactly what is happening in this case for each address there is a delay, but since the delay is are overlapped, the idea is overlapping since the address this is the delay associated with these address is overlapping, we could get a pipelined transfer of data once. So, the timing of the signals should be in such a way that by the time D 0 reached that the controller, D 1 should be ready to transfer.

So, by the time D 1 is reaching the other side of the transmission D 2 should be ready. So, the appropriate signals, which generates D 0 D 1 D 2 and D 3 the corresponding data from the row buffer has to be carefully timed. So, under this context the timing parameter associated with DRAM is a very crucial design issue.

(Refer Slide Time: 15:18)



Now, let us try to see how address mapping is done. Consider the case we have a single channel, let us say we have an 8 byte memory bus; that means, the memory can transfer 64 bits of information that is why it is called an 8 byte memory bus. And, you have a 2 GB memory, which is consisting of 16 K rows and 2 K columns per bank and you have 8 banks in total.

Since, it is 2 GB the total address bits would be 31 bits. And, you can have different types of interleaving; first one is called row interleaving so, what do you mean by row

interleaving? Constitutive rows of memory is kept in consecutive banks. So, when you have 1 fold row when you move to the next row, the next row is not part, for example, row number 200 and 10 210 is in bank 0, 200 and 11 will be in bank 1, 200 and 12 is in bank 2.

So, when you have a couple of request, which are 2 adjacent rows these are not stored in or the addresses are mapped in such a way that, they are not stored in the same bank. Row number 200 and 10 will be in this bank, row number 200 and 11 will be the next bank.

So, when you have a big copy of data, where the data is scattered across 210 and 11, if they are belonging to different bank by the time the transfer is over in 200 and 10, 200 and 11th data can be kept ready. So, accesses to consecutive cache blocks are serviced basically in the pipelined manner. So, you can see this out of the 2 GB, that you have we have the 8 byte memory bus. So, the last 3 bits will tell byte in the bus and we have seen that 2 K columns so, 11 bits are given to the column.

And once these 11 bits all permutations of these 11 bits are over, now we are moving to next row. And, we have seen that consecutive rows are belonging to consecutive banks. So, once your column bit is over the next 3 bits we have total of 8 banks so, the next 3 bit is coming for the banks and then we have the rows that is coming.

So, in short the bank which will come between the row bits and the column bits this is called row interleaving. So, once all the column bits are been given when we are moving into the next row, the bank bits will come over there. Yet another kind of interleaving is called cache block interleaving, what is the significance of cache block interleaving? We know that, it is a last level cache that is going to request to the DRAM to give a block of data.

So, any data that moves from DRAM is to the processor chip, especially to the last level cache. So, it is not other one word that you want you wanted to transfer an entire cache block to the last level cache from the DRAM controller. And, once that is given once CPU is going to fetch from those cache blocks; now immediately you require the next cache blocks.

So, once you transfer a block of cache from the DRAM in the near feature the next block will be of demand. So, can I keep the adjacent cache block data in adjacent banks so, consecutive cache blocks addresses is stored in consecutive bank that is you have 64 byte cache block, access to consecutive cache block should be done in parallel. So, this the way how it is being done?

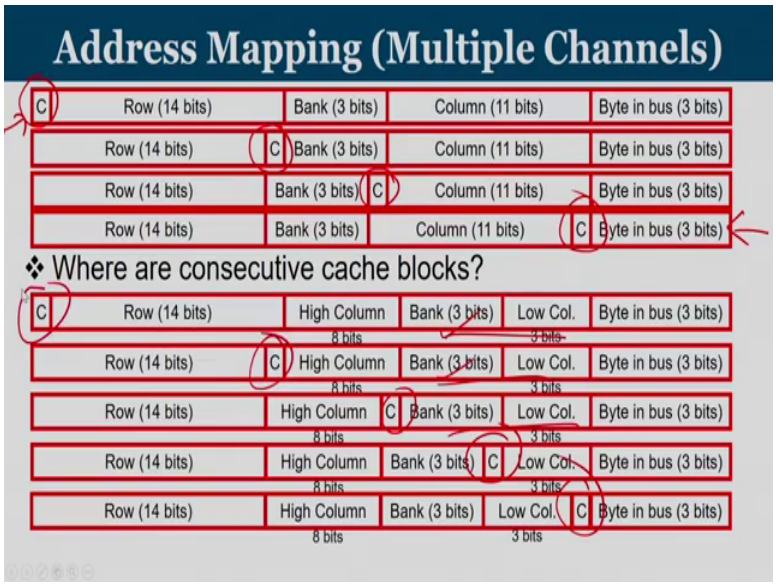
So, when you have cache block interleaving, you will still use the last 3 bit belong to bus because you are having an 8 byte memory bus, your column bit was 11. Now, we know that how much bits are required to generate the data that is correspond to 1 cache block. Our cache block is 64 bytes so, we required total of 6 bits combination of 6 bits starting from 0 0 0 0 0 0 to 6 1's 1 1 1 1 1 1 this is 6 bits are there you 2 power 6 combination and that is what is your 64 bytes.

So, the lower order 6 bits, we determine the contents of a cache block. So, the next one these things will go to one cache block, the next very next address is next cache block. Can I put banking there that is why bank bits come after the last 6 bits, why last bits? Because, you have a 64 byte cache and 64 byte correspond to 2 power 6. So, least significant 2 power 6 bits are there, after that you use the bank bits and then which is followed by the remaining column bits we know that the column bit is 11 out of it is 3 is used, the next 8 is being given.

So, what happens in cache block interleaving is out of the total column bits you are going to fragment the column bits and insert the bank bits inside this. So, depending on your requirement, let us say you want raw interleaving use the address mapping accordingly. So, the DRAM controller can play this a bit of intelligence on it and give the addresses in such a way that will suit our requirement.

So, if row interleaving is needed, then bank bits should be kept in between the row address bits and the column address bits. If, cache block interleaving is required, then you have to split the column bits, you have a lower order column bits and the higher order column bits. The least significant n bits where 2 power n is going to be your cache block size the least significant n bits are been removed and the bank bits will come just after the least significant n bits. So, this is basically the address mapping in a single channel.

(Refer Slide Time: 21:23)



Now, let us see address mapping in multiple channels. So, when you have multiple channels let us say you have 4 channels. So, 2 bits in the physical address, they are going to take a call whether it belong to channel 0, channel 1, channel 2 and channel 3. So, we can see that if the most significant bits are been used for channels, then after this byte column banks and all you have this channel also that is going to come. Now, I can keep in such a way that challenge will can be kept after the column and bank bits.

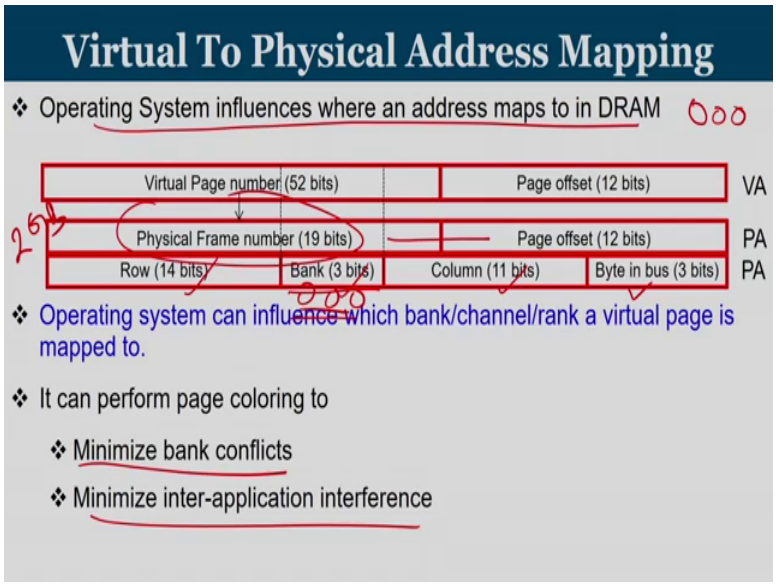
We can also keep channels in between banks and column bits or this the last 3 design, where adjacent words are kept in adjacent channel. So, once this channel bit is located between column bits and the byte in the bus bits the last case, this means adjacent memory words are kept. So, your word number 0 is from channel 1, word number 2 sorry word number 1 is from channel 2, word number 2 is from the next channel, that is why if the channel bits is put immediately after the bus.

Now, once you go to columns then the next 2 column so, you have done with all the column bits the next row is going to be in adjacent column. Next bank is going to be an adjacent channel, next the most significant case wherein the entire system is divided into multiple channels and the peculiarity is adjacent location for kept in the same location. So, the first 1 by n portion will be in one channel, next 1 by n is going to be other channel. Now, along with this if you have wanted cache block interleaving how will you

do here also we can see that, for a 64 bit cache blocks you can see the banking bits are still there.

So, the lower order bits you have seen it already, now in between the channel bits can be added if cache block interleaving is to be applied. So, row interleaving and cache block interleaving are the 2 common method of a using this banking and depending on whether it is a single channel or multichannel appropriate channel bits also has to be given.

(Refer Slide Time: 23:33)



Now, let us try to understand the very important concept, which is very much needed while designing of the DRAM system. We know that our processors are going to generate virtual address and using memory management unit you convert this virtual address into physical address.

Physical address is the address of the location when it is residing inside main memory. Now, main memory is what we deal as DRAM? Now, while you give address it is operating system that gives the address via a page table or a memory management unit.

So, when you translate an address from a virtual address to physical address, whatever frame number you give that has a significant say whether banking will happen or interleaving will happen. Consider the case you have a virtual address or the operating system in short can influence, where an address maps into a dram. So, your meaning a virtual page number and using a translation look aside buffer or a page table virtual page

number is converted to physical frame number. So, you consider this case you have 19 plus 12 that is total of 31, you are going to have a 2 GB of RAM, that is why it is 31 19 plus 12 it is 31 bits.

So, from our 64 virtual address we are going to convert into 31 bit virtual address. So, essentially, it is this 31 bits that consists of your frame and offset is to be divided into byte in bus column bits, bank bits and row bits. So, whatever physical frame number you have total of 2 power 19 frames that is why you use 19 bits for frame number. So, when somebody is requesting for a physical address; that means, you are going to assign a physical address for an incoming virtual address, that 19 bit physical frame number that you give we will take a call what is a bank number?

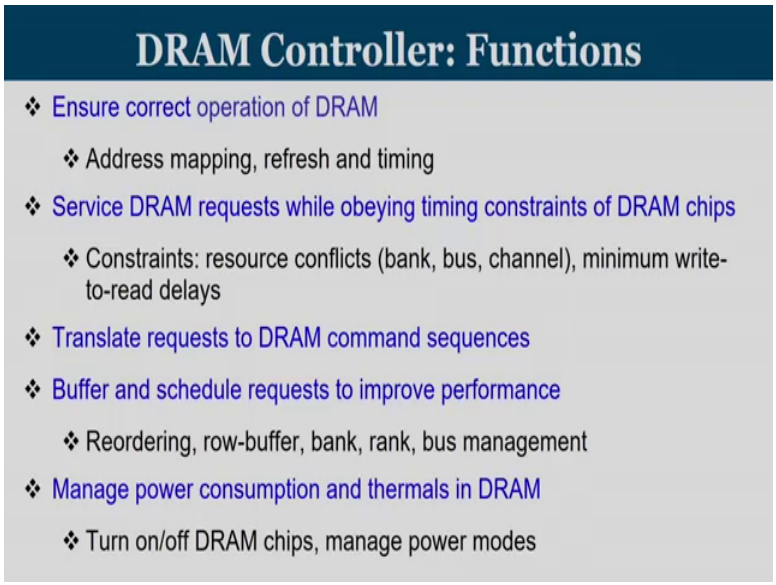
Let us consider that a program is always running parallel with some other program. Let us say a and b are 2 programs. Now, a is already residing; now you are going to lowered program b from secondary memory into main memory. What will be the address that you give to b? You do not want the conflict, you do not want the bank conflict between a and b, because condense from program a and condense from program b has to be fetched to more or lesser the same me time they are co running programs or concurrently running programs.

So, when you assign the physical address to be the bank number that you give to the physical address of b should be different from the bank number of the physical address of a. Assume that I have given a bank as 0 to a. So, when I give it tob, b should not be given a bank address of 0 and how will I know bank? You have to give that frame number such that these 3 bits are not equal to 0, when those 3 bits of the frame number is equal to 0 0 0, then program a and program b is going to have a banking level conflict.

So, how it is being done? Operating system can influence, which bank or channel or a rank a virtual page number is mapped to. It can perform page coloring to minimize bank conflict or to minimize inter application interference. So, if you know certain applications are going to interfere each other, if you want them to be either in the same bank or in the different bank, the moment you give the virtual page number to physical frame number translation, it is during the address assignment operation you have to take care of this.

So, operating system has a big role to play the underlined physical organization of the main memory has to be connected with the operations inside the DRAM. So, we have completed the address mapping. Now, we will see what are the functions of a DRAM controller?

(Refer Slide Time: 27:26)

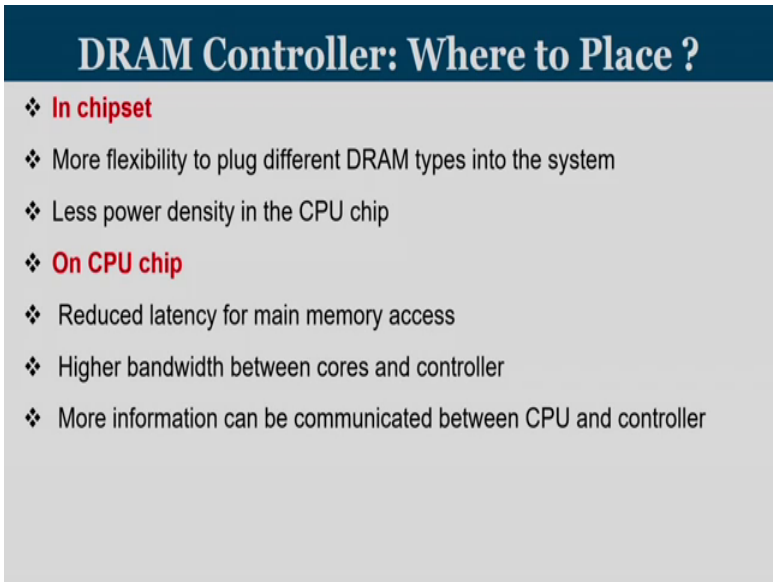


So, DRAM controller does many functions first one is it has to ensure correct operation of the DRAM like, address mapping, refreshing and timing. The service DRAM request whatever request is coming to the DRAM, the DRAM controller has to service them by using the appropriate scheduling principles. Principle you have lot of constraints you will get like the resource conflict, sometime the request will be to the same bank, sometime it may be to the same channel, sometime you have to make use of the bus.

So, you have to minimize the delays and whatever request you get you have to translate these request to do command sequences that the DRAM device can understand, we have to buffer these request and from the buffered request like called queuing, you have to pick up certain request that is called scheduling schedule the request. And, sometimes once you get multiple request which one to pick that is called DRAM scheduling. Scheduling is also an important operation, sometimes you may have to reorder the service that you get reordering; row buffer management, bus management, rank management are there.

And therefore, what we have something like management of power and thermal aspects of DRAM. DRAM is going to consume power. So, if I schedule in certain way then probably I could get a better power efficiency. It is done with the help of turning off of turning off certain DRAM chips or moving into certain specific modes in DRAM. So, there is a the same choice, which is not there now, but a couple of years before there was a design choice where should I keep the DRAM controller.

(Refer Slide Time: 28:53)



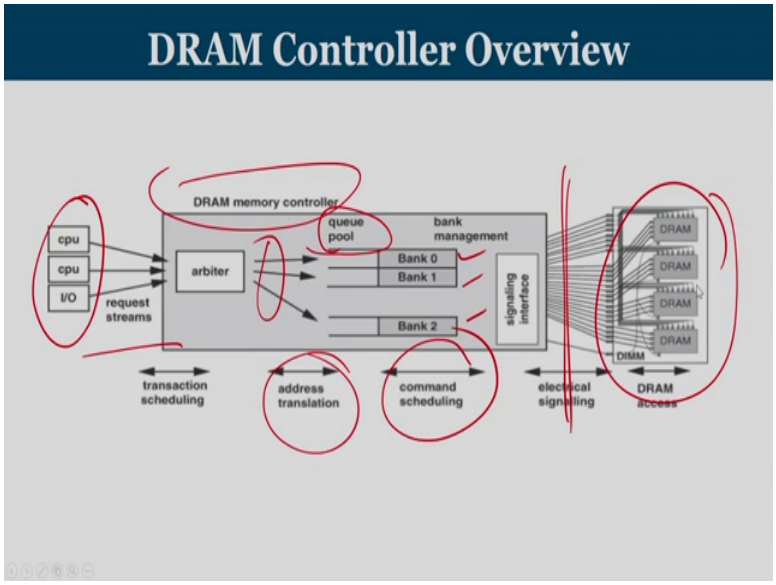
It can be placed on the chipset there is on the motherboard. So, there is more flexibility, like if use a different motherboard you can use a different DRAM controller, because processor and motherboard or two separate entities. So, if I wanted to use a different DRAM controller, you would always have this processor in a different set of chipset.

So, more flexibility you have to plug in different DRAM types into the same system. And, especially it is less power density in the CPU chip since, DRAM controller is not part of CPU chip and DRAM controller is part of the chipset the motherboard, it we to reduce power consumption inside CPU.

But, nowadays it is preferred that is since there are lot of transistors available inside the CPU, DRAM controllers can be kept on the CPU chip possible. This will reduce the latency for main memory access, because it is inside the chip, higher bandwidth between cores and controller and CPU and the controller can talk more of frequently. That means,

more amount of data, can be exchanged between the CPU and controller, when they both are located inside the CPU.

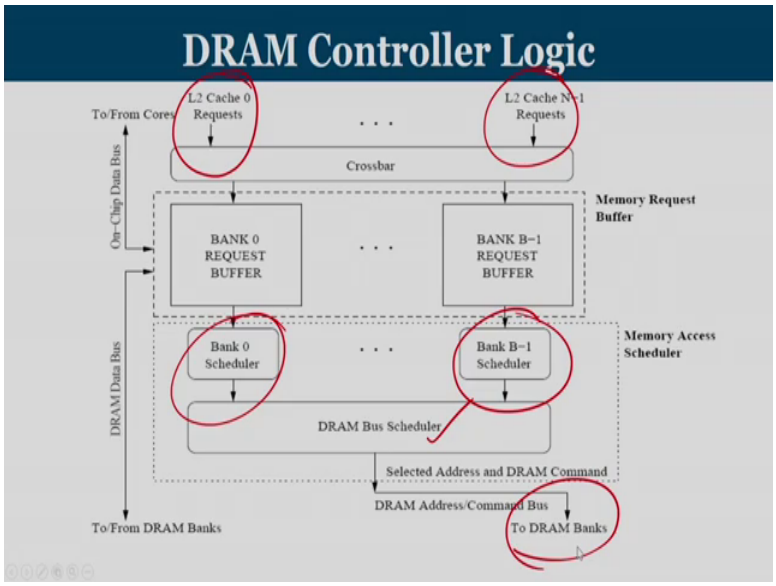
(Refer Slide Time: 29:59)



This is a logical view of the DRAM controller we can see that CPU and I O devices are going to view the request and this is the DRAM memory controller, you have various banks. So, request various banks have to be signal the appropriate address translation so, we have different request that is coming you performed address translation.

Once you perform the address translation we will come to know, whether this request belong to bank 0, bank 1 and all and each bank we are going to put up into a queue. And then you have the scheduling out of the many a request that comes to bank 0, which is the one that is to be picked and then you have to signaling interface we should essentially work with your DRAM. So, this is the logical view of the DRAM controller.

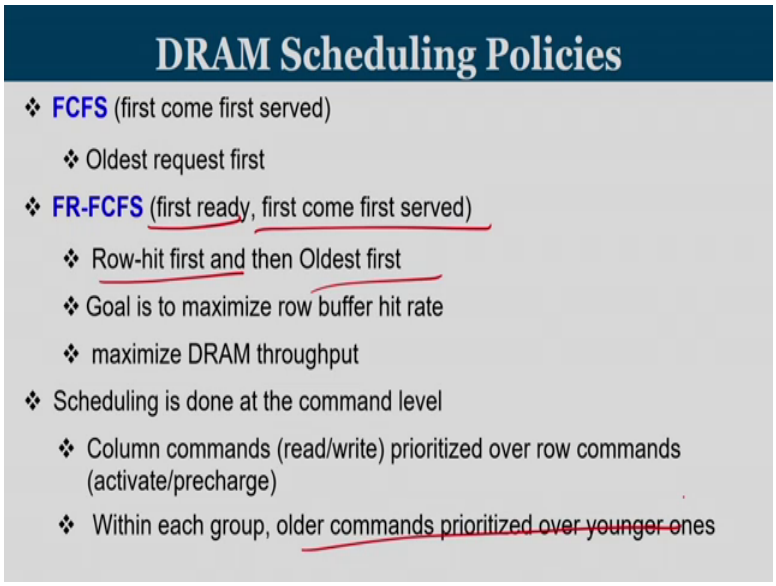
(Refer Slide Time: 30:41)



We can see that, when you have multiple caches, multiple last level cache, all these are going to give you request and then you have each of this banks. So, this request will go to the appropriate banks and there is a scheduling is there and scheduler will pick and activate the necessary banks.

Now, what are the different DRAM scheduling policies? So, what is essentially scheduling? When you have multiple request that is coming from the multiple request can you pick one, this picking is a process is called scheduling. Now, what are the parameters by which a pick that is important?

(Refer Slide Time: 31:21)



The simplest one is when you get multiple request pick the one who reads the DRAM queue first, First Come First Served FCFS Scheduling. So, oldest request (Refer Time: 31:30) is given preference, the second one is known as FR- FCFS; First Ready, First Come First Served.

So, one who know here is you know that, you are going to activate a data by giving a column number. So, that the condense of the entire row will come to a row buffer and then you give you the column address, such that the data can be extracted. Now, if the next request is also to an already opened row, then again row address need not be given, because the row is already present in the row buffer so, just give only column address.

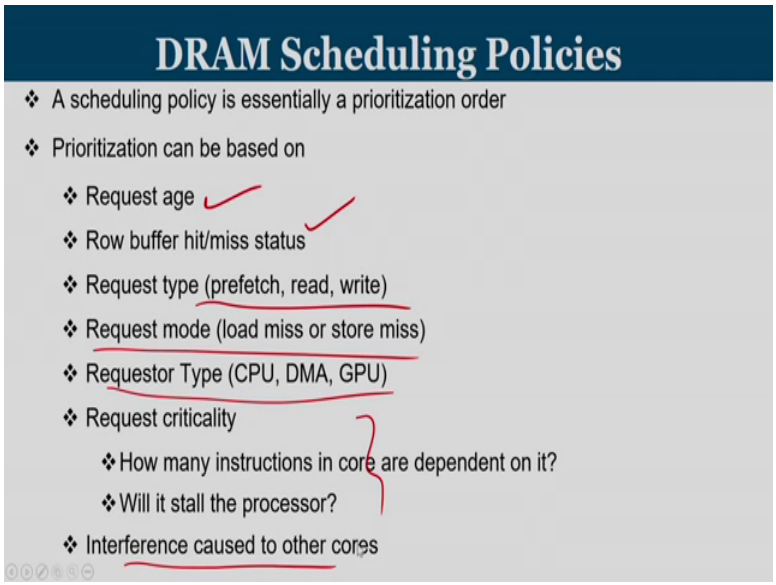
So, if you can give old request who are row hits; that means, there condenser already present in the row then it is very easy so, that there is not much delay that is involved. So, the idea is very simple, when you get many a request find out from this set of request, which of this request is to an already opened row. That is a row is already ready so, first service those request which are part of an already opened row that is known as first ready and between them you can use first come first serve.

So, row hit is serviced first and then only we are going for oldest hit. So, certain packets or certain request that are coming to DRAM controller, which are actually part of some different row may not get service. So, the olden request may not be part of a currently opened row. So, in that case oldest one may not be preferred a later request the request

which comes just recently, but if that is a row hit we can prioritize that, such a kind of a scheduling policy is known as FR-FCFS. This will surely maximize DRAM throughput, because rather than writing and then lading it again, already you can gives those one which are part of row hits. And, scheduling is basically done at the command level we know that for you have to give once scheduling is done and the appropriate commands are also been passed.

Column commands are always prioritized to our row commands. So, whenever there is a chance that we can give only column commands, we should give that because, once you give the row command then the appropriate row has to come down to the row buffer and then only we can give the column commands. Within each group older commands are prioritized over the younger ones.

(Refer Slide Time: 33:52)



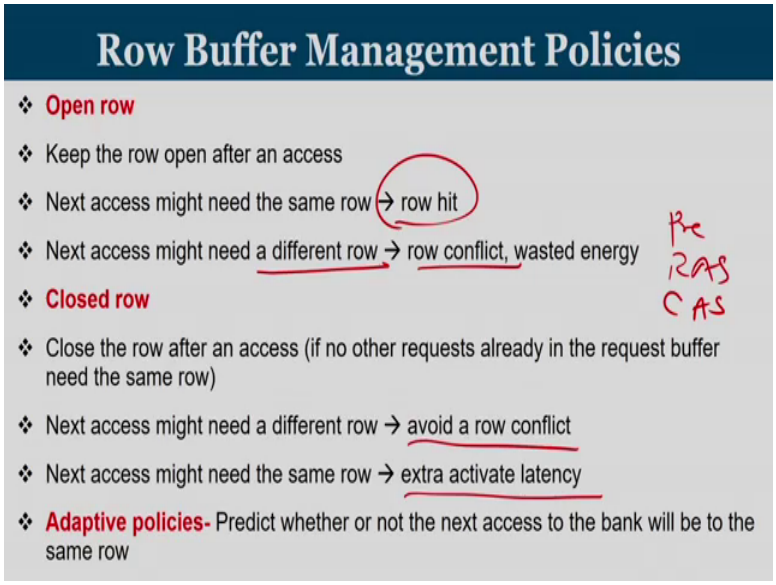
Continuing with the policy a scheduling policy is essentially a prioritizing order. And, what are the method by which you can prioritize, you can look at the age of the different request that is coming, you can look whether from among the request some of them will be row hits, some of them will be row miss, you can see the request to DRAM will be coming from a cache controller.

So, whether it is a prefetch request or a read request or a write request, some are load misses, some are store misses, some of the request coming from GPU, some are coming from CPU, some are coming from DMA controller so, there also you can use some

priority. And, you have to look into how much critically is this request? How many of the instructions are actually dependent on this particular DRAM request? If, I service this request early, how many instructions are going to get benefited is the processors installing?

So, since you working multi core processors, many processors many caches are there from the caches there is possibility that multiple request will come to the DRAM controller. Once you get multiple request coming to the DRAM controller, how are you going to pick them? And, this picking can be based upon age row buffer hit or miss status, whether it is a prefetch operation or read or write operation, whether it is a load or store miss or who is going to give the request. Let us say it is a CPU, DMA or GPU and the criticality of the request. Sometimes the interference caused by these requests are also being considered in order to take the final call.

(Refer Slide Time: 35:34)



Now, row buffer management policies, here we are going to deal with there is a case like we have an open row. So, when you get a request and you find from the address translation in such a way that the row number of the incoming request is same as the adopt the row number of the currently activated row, that scenario is known as open row.

So, if you use open row policy; that means, even after servicing a request you are not going to close the row. You assume that the next request you feel that or the you the probability that, the next request also belonging to the same row it is very high. In short

when we work with open row policy, after accessing a row the row is kept opened, hoping that the next request also will be to the same row. If, the next request happened to be to the same row, then this row address strobe is not required already it is an opened row just give column address and get things done easily.

So, next access might need the same row so, if the next access needs to the same row that is known as a row hit, if the next access by chance unfortunately, if it is to a different row it is a row conflict, at the time I have to precharge, then I have to apply a row address strobe and then I have to apply the column address strobe. The, second policy is known as closed row policy, in the case of a closed row policy, what we do is close the row after the access. So, for every access you close the row by applying a precharge signal.

So, once you use a precharge signal the condense of row buffer has transfer back to the row hoping that the next request will be to a different row. If, the next request is to a different row already you have done the precharging so, only a row activation is required. So, close the row after an access, if know other request is already there in the request buffer.

If there is one tricky portion is after accessing of a row the condense of the row by giving an appropriate command, if you have an address you look into the row buffer and transfer the data. After transferring of the data you look into the DRAM queue, whether there are any request to the same row is so, service them, if they are no other request to the same row or the queue is empty kindly pre charge it if you precharge it is now precharging is over row buffer is empty.

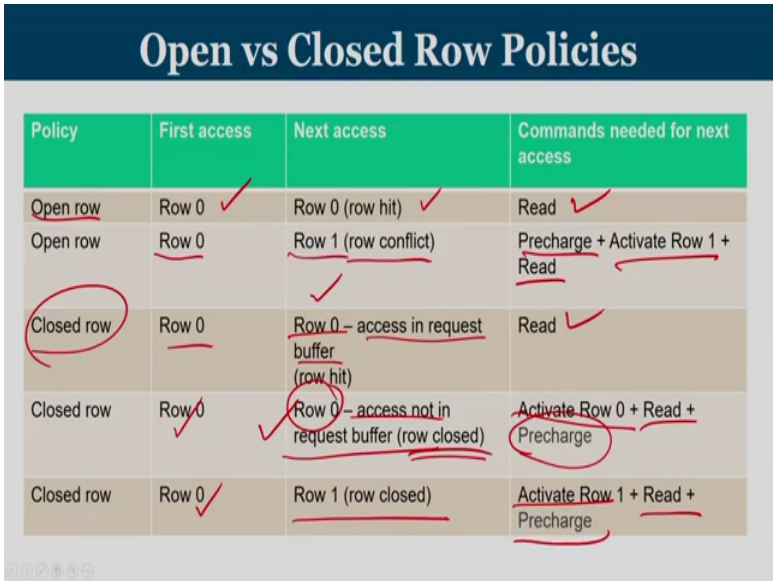
So, any request that comes in future you have to surely give a row address. So, next access may be needing a different row. So, you avoid a row conflict, it is just like a row empty case. If the next access it to a same row, then you have to again activate it, this is the basic difference between an open row policy and a closed row policy. Open row policy believes that the next request will be to the same row so, if that is a case that it become a row hit.

If, it is to a different row then you have to precharge it, then row address strobe and column address strobe. Coming into close row policy, the concept of closed row policies in is like this, we assume that the next request is not to the same row. If, it is not to same row it will surely a row conflict so, you have to precharge at that time, we precharge it

now itself we are not waiting for the next request to come prior to the next request we precharge it already. If you precharge it whenever is as go as like row is empty now for every new request that is coming you have to activate the row.

And, nowadays we use adaptive policies certain applications may be opened row friendly, some other applications may be closed row friendly or in the application certain time phases up to 100000 cycles, open row policy may be good and after 100000 cycles closed row policy may be good. So, depending on the applications requirement you have to understand what kind of policy will suit them? So, now a days there are research work which look into the profile of the application and from the profile of application see whether open row policy suitable or closed row policy suitable.

(Refer Slide Time: 39:51)



So, we can see there is a near comparison between open and closed row policies. Let us say you are using an open row policy the first access is to row 0. Let us say the next access is also to row 0 since it is an open row policy, it will be always a read we have only just give the read command it is kept open. The first access is to row 0 and second access is to row 1 it is a row conflict so, you have to precharge first activate row 1 and then only you can read. Now, assume you are using a closed row policy let us say the first access is to row 0, next access is also to row 0 access in request buffer.

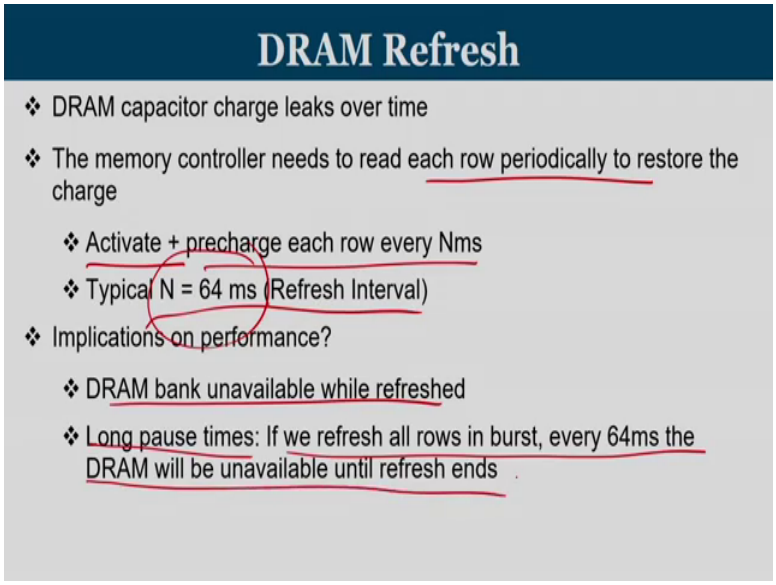
So, if this next access is already present in the row buffer. So, by the time the first access is over you look at the queue you see that there is somebody there and he is also to row 0

just read. If, at the time the queue is not having that one, then we will close it. So, if it is close you have to activate row 0 again; that means, when the first access is to row 0 the condenser extracted, after extraction when you look into the buffer that buffer does not have any more request; that means, the second request is not buffer till then so, then the row is closed.

After closing only we will get the new request in that case you have to activate the row and then read once reading is over at the end you have do a precharge. The third case is in the closed row policy the first access is to row 0 the condense are (Refer Time: 41:25) the next access is to row 1.

So, you have to activate row 1 read at the end of the operation it is precharge. So, precharge happens after every access. Now, we know that since a DRAM is making use of capacitors and capacitor has a leakage property. If, you read from capacitor it will lose the charge, even if you are non-reading it is still lose the charge may be it may thick little bit more time, which is what is known as the leakage property of the capacitor.

(Refer Slide Time: 41:55)



So, we will see what DRAM refresh is? DRAM capacitor charge leaks over time, the memory controller needs to read each row periodically to restore the charge. So, what is the idea of refreshing? You have many rows let us say you have 10 rows you activate row number 1.

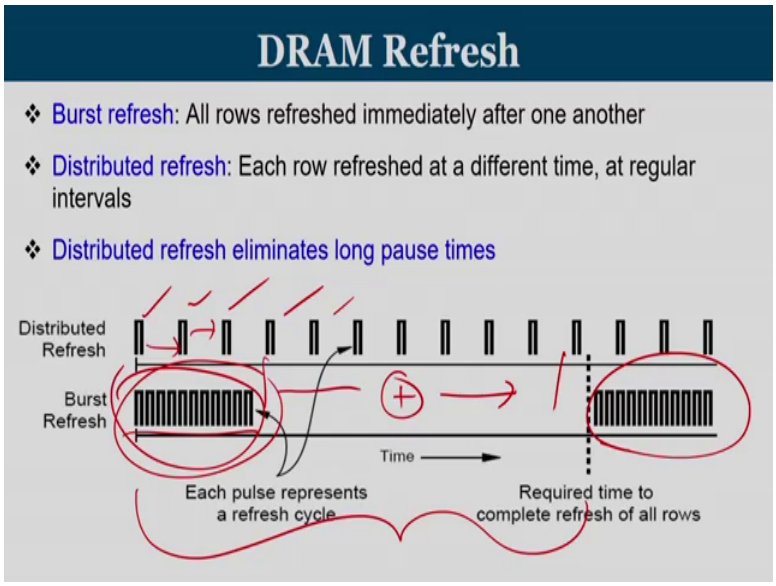
So, the contents of row number 1 will come down to row buffer, then you precharge it so, it is stored back. So, it is a sequence of activate it the row condense will reach row buffer precharge, it back it will goes back to the row buffer take the next row, activate precharge go to the next row activate and precharge. This has to be done, periodically such that the charges that is stored in the capacitors of the appropriate rows are not decade.

So, memory controller needs to read each row periodically to restore the charge. So, activate precharge each row at every N millisecond, where N is typically the refresh interval. So, this refresh interval 64 millisecond means, in every 64 millisecond a DRAM cell has to be refreshed; that means condense has to be transferred into the row buffer. And, then store it back and this has it is own implications on performance meaning, when the DRAM is in the refresh operation then it is not available for outside request, DRAM bank is unavailable during the refreshing operation.

So, during refreshing it takes some amount of time to refresh all D rows. So, you have long pause times if we refresh all rows in a burst at every 64 millisecond DRAM will be unavailable to cater the request. So, refreshing is basically an operation in which an entire row is transferred to row buffer and then it is stored back. Once, you store it back all the capacitors are properly charged representing whatever data you wanted to store in, but since you have so, many rows all the rows has to be taken one after another and during this refreshing operation DRAM is not available for outside request.

So, we experience a pause DRAM will service request then it goes for this internal mechanism of refreshing, during that time no other request for entertained. So, DRAM only available for sometime during it is the request happens again it is available again for a shorter time it is not available.

(Refer Slide Time: 44:24)



So, DRAM refresh are of 2 types; 1st one is called burst refresh all rows are refreshed immediately one after another take one row or refresh it take the another row refresh, it do it for all the rows and then DRAM is opened for others. Again, after sometime again you move into the refresh 1. The other one is called distributed refresh each row is refreshed at a different time so, you take at the first row refresh then DRAM is opened for normal read write operation, after some time you go and refresh second row.

So, the pause time is very less each has it is own merits and demerits distributed refresh eliminates long pause times. So you can see from this graph, if it is a burst refresh let us say these are the rows each row is refreshed and this much is the time by which the row has to be refreshed again. So, during this time from this point to this point DRAM is available. Once the time period is over you have to refresh all the rows again.

So, you have the burst of refreshing that happens. Let us say there are 10 rows, now in this new case all these 10 rows will be refreshed not like in a bursty mode you refresh only 1 row at a time remaining time DRAM is available, here also DRAM is available this is called scattered refresh, but we can see that whenever the refreshing is in progress DRAM cannot service the data accordingly.

So, burst refresh and distributed refresh are the 2 mechanisms by which, refreshing happens inside DRAM. So, with this we come to the end of this DRAM structures so, just to summarize what we learn today. We are trying to see how address mapping is

done, because we know that DRAM consists of channels DIMMs ranks banks rows and columns. So, given an address which bit will represent the rank, which bit will represent the bank, which is for rows and columns.

And, we have seen there are different types of interleavings and basically your banking is going to help in paralyzing things. So, where should be the bank bits coming row interleaving and you have cache block interleaving. And, then we have seen when there is multiple channels, where will be the channel bits be adjacent words can be in adjacent channels, adjacent words, when it goes to adjacent channels. So, word number 1 you fetch from 1 channel word number 2 you fetch from another channel like that.

Similarly, adjacent columns adjacent rows, adjacent banks, can be in adjacent channels. Moving further, we have seen what is the role of operating system, when it gives a physical address or a physical frame to a virtual address that has an implicit control over what is a bank that you are going to give? We have learned the functions of DRAM controller, what it does the DRAM scheduling policies and then we concluded today's discussion with the refreshing operation of DRAM.

So, with this we complete 10 lectures of the course. The course is divided into storage aspect and the interconnect aspect, now our pipelining aspect is covered, our cache memory aspect is covered, our main memory aspect is covered. And, now we are fit to learn try to understand, what is multi core and how this storage mechanism what we have learned is going to help in improving the performance of a multi core processor? I hope you are enjoying this course we are almost half done another 10 lectures more. Feel free to contact us if you have any queries.

Thank you very much.