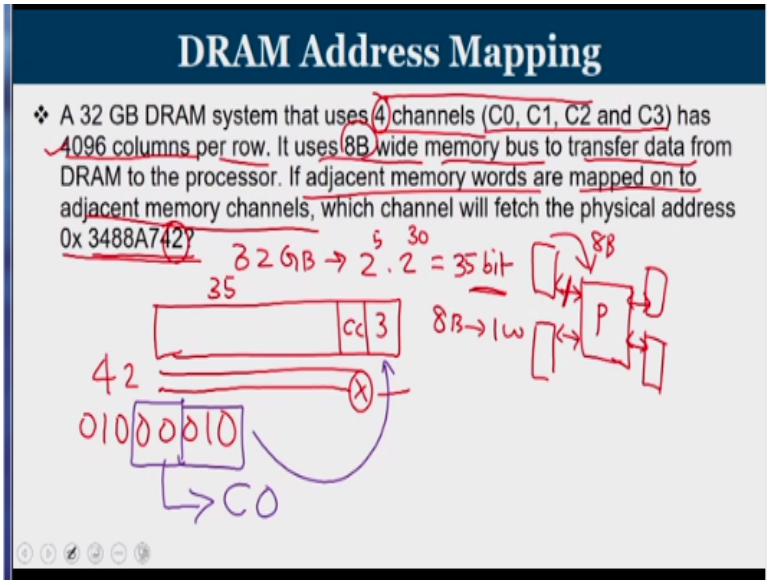
Advanced Computer Architecture
Prof. Dr. John Jose
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology-Guwahati

Lecture-28
Tutorial Design Concepts in DRAM and HardDisk

I welcome all of you to the 9th tutorial, in this tutorial we work out few problems with respect to the design, address mapping and working of DRAM as well as hard disk. The concepts that we have learned in the lecture videos will help you in solving these problems. So I request you prior to attending this tutorial video kindly go through the lecture videos of DRAM as well as the hard disk.

(Refer Slide Time: 00:57)



The first tutorial question for the days let us consider a 32 GB DRAM system that uses 4 channels C0, C1, C2 and C3 it has 4096 columns per row. It uses 8 byte wide memory bus to transfer data from DRAM to the processor, if adjacent memory words are mapped onto adjacent memory channels, which channel will fetch the physical address that is been given. Let us try to understand what is the concept of channel, so let us say this is the processor.

Now your memory is connected through 4 channels that is a meaning of this, so here we have different DIMMs in DRAM and this 4 channels basically means there are 4 point of entry from

the DRAM system into the processor. So this permit us to parallely access 4 memory locations provided they are from different channels. So in this 32 GB DRAM, so 32 GB that corresponds to 2 power 35 that is 32 into GIGA stands for 2 power 30, so we have 35 bit address.

$$32\ GB = 2^{35}$$

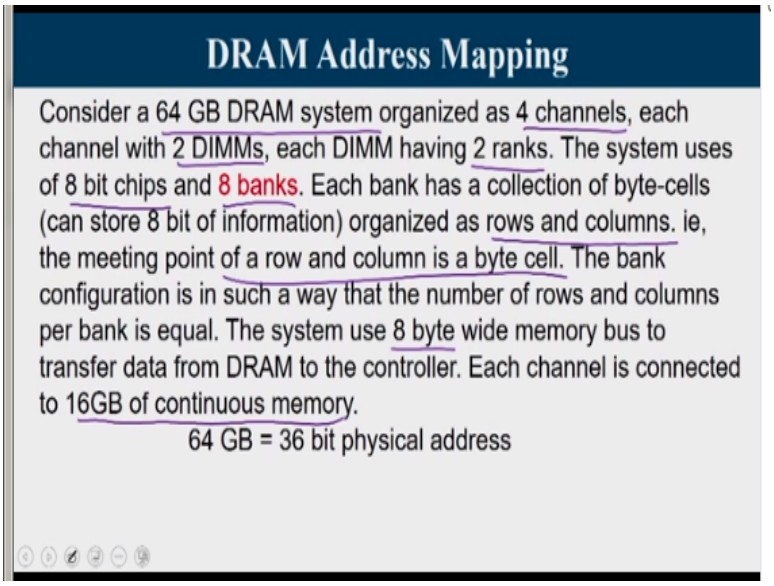
Now this 35 bit address whatever we are talking that has now it is still there are 4 channels. So some of the bits then there are 4 channels 2 bits, somewhere in the address will tell you which is the channel this particular address is mapped to. Now this was 4096 columns per row and it uses 8 byte wide memory bus, so when you bring something from the DRAM into the processor, it is 8 byte is coming together these are 8 continuous bytes in the memory.

So the last 3 bits in that address they belong to 8 continuous bytes. So whatever be the most significant 32 bits you keep them constant. The last 3 bits only if you vary, these are the bytes that are brought together in one fetch. Now if you are telling, so this is what is called a word, so these 8 bytes constitute your one word. The peculiarity of the address of the bytes within a word is they all have the address the most significant 32 bit of the address is same.

The bytes of a single word will differ only in the last 3 bits of the address. Now if adjacent memory words are mapped onto adjacent memory channels, so adjacent word means let us say this is one word, the very next word will all this be same except this bit, that is immediately before the last 3 bits that is a word that will differ that is called adjacent word. If adjacent words are mapped on to adjacent memory channels, so these 2 bits will be your channel bits.

Such that the adjacent words are being assign to adjacent channels, so given in this address this is the address that is been given let us take a 42. Now this 4 if you write 0100, this 42 is the last 8 bits, now 2 is 0010. Now these are the last 8 bits out of which the last 3 bits will be part of byte within a bus and it is these 2 bits that will tell you which is a channel. So this indicates that this particular address is mapped to channel number C0 that is been given in the question.

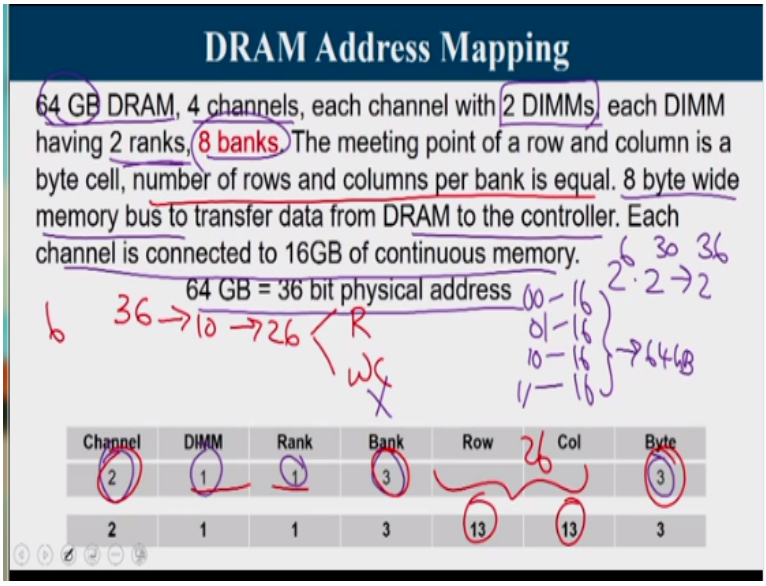
(Refer Slide Time: 04:43)



Let us now move into the next question in DRAM address mapping, consider a 64 GB DRAM system organized this 4 channels. Each channel with 2 DIMMs, each DIMM having 2 ranks the system uses 8 bit chips and 8 banks, each bank has a collection of byte-cells which can store 8 bit of information organize thus row and column. That is the meeting point of row and column is a byte-cell. The bank configuration is in such a way that the number of rows and columns per bank is equal.

The system uses 8 byte wide memory bus to transfer data from DRAM to the controller, each channel is connected to 16 GB of continuous memory. Let us try to rephrase what are the things that is been given in the question the important points.

(Refer Slide Time: 05:31)



So in this case, it is been told that we have a 64 GB DRAM, 4 channels, 2 DIMMs. Since it is 64 GB, 64 GB means 2 power 6 into 2 power 30 that is 2 power 36 is a number of bytes you require 32 bit address.

$$64GB = 2^{6} * 2^{30}$$

So 64 GB is 30, so you require 36 bit address 2 power 36 bit, so 36 bit physical address is there. Now this 36 bits we have to divide into channel DIMM, rank, bank, row, column and byte now it is an 8 byte wide memory bus like in the previous question.

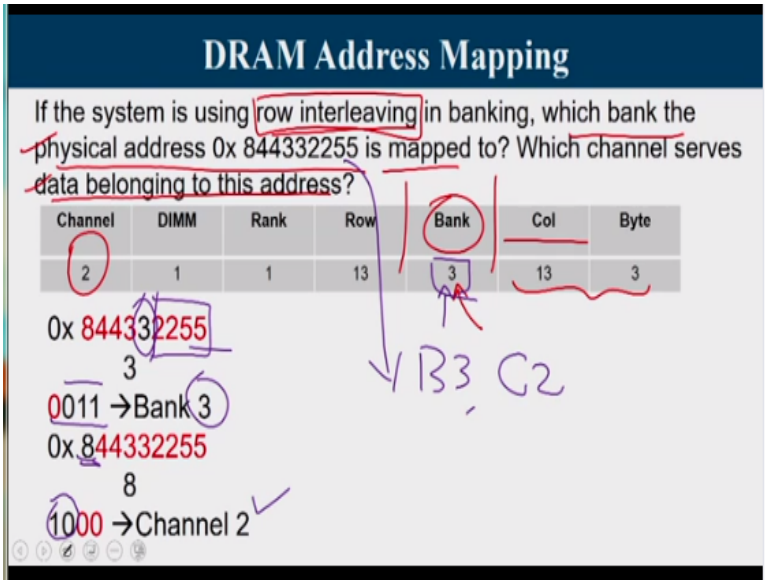
The last 3 bits in the address will represent byte within a bus is used to transfer from DRAM to the controller. Now each channel is connected to 16 GB of continuous memory when you have channels wherein the 16 GB of continuous memory are part of the same channel. So it is a first 16 GB is 1 channel next 16 GB, third 16 GB and the last 16 GB together they make out 64 GB. So if this is the organization, they should have differ only the most significant bit, if it is the most significant 2 bit is 00, it is a first channel.

Then 0110 and 11 that is why the first 2 bits are channels, now let us say we have 2 DIMMs. So somewhere 1 bit is therefore the DIMM and generally it will be there the most significant bit immediately after the rank. And we know that DIMMs are further divided into rank, so you have 1 bit for rank, we have total 2 ranks. So 1 bit DIMM and 1 bit rank and then we have 8 banks, so somewhere we have 3 bits inside the address.

So this location of the bank is still not sure for you anyway there are 3 bits that is been reserved for the bank. So out of 36 we have 3 bit for the byte within the bus that is what this is talking you have 2 bits for channel that make it 3 + 2, 5 bits is already gone. And then we have 3 bits for bank that make it 8 + 1 bit each for DIMM and rank, so out of total 36 bits already 10 bits are gone. So remaining you have 26 bits, these 26 bits I have to split into rows and columns per bank.

Now it is been mentioned that the number of rows and columns for bank is equal, so the remaining 26 bit whatever we have, I will divide into 13 bit row and 13 bit column that is what is been given.

(Refer Slide Time: 08:03)



Now if the system is using row interleaving, so these row interleaving will tell you where you how to keep it in the bank, which bank the physical address 0x844332255 is mapped to, which channel serves data belonging to this address. There are 2 question one is which bank this address is map, second one which channel it has been mapped. Now how will you find out bank, so since you are using row interleaving we have to find out where the bank location are, row interleaving means adjacent rows are mapped to adjacent banks,

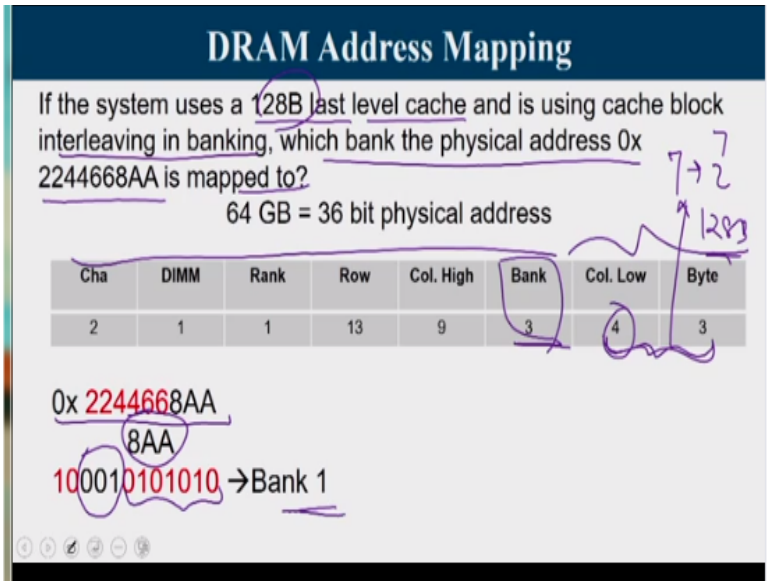
So you know that you have rows and columns, so once all combinations for the column bits are been given then we are going to change the row. So once you completing of a row means you have given all combinations of columns, so it is after the column bit. So when you apply row interleaving as discussed in our lecture video, the bank bits will come in between row address and column address.

So these are the 3 bits for the bank and we know that the most 2 bits the most significant 2 bits that is been given to channels. So given the address 844332255, you know that the last 16 bit is gone for column number and byte. So I am removing the last 16 bits that is not interest of to me the next 3 bit, so this represents a 4 bit 3, so the binary value of 3 is 0011 out of which you take these 3 locations, so that will tell you 011.

So that means that this particular address is mapped to bank number 3, so let us try to conclude what we do here given the address, we find out what are the bits for channels and banks since it is a row interleaving the bank bit should be in between row and column. So you have to extract these 3 bit from the given address, so this is a 36 bit address out of it is these 3 bit is of interest to us. So we remove the last 16 bit take off the next 4 bit that defines 0011 from that you take up the 3 bit that is bank 3.

Now if you wanted to look at the channel it is a most significant 2 bits, so the most significant hex that is been given is 8, 8 means 1000 and these is the most significant bit that tells that it is channel 2. So this particular address is mapped to bank 3 and then channel 2.

(Refer Slide Time: 10:22)

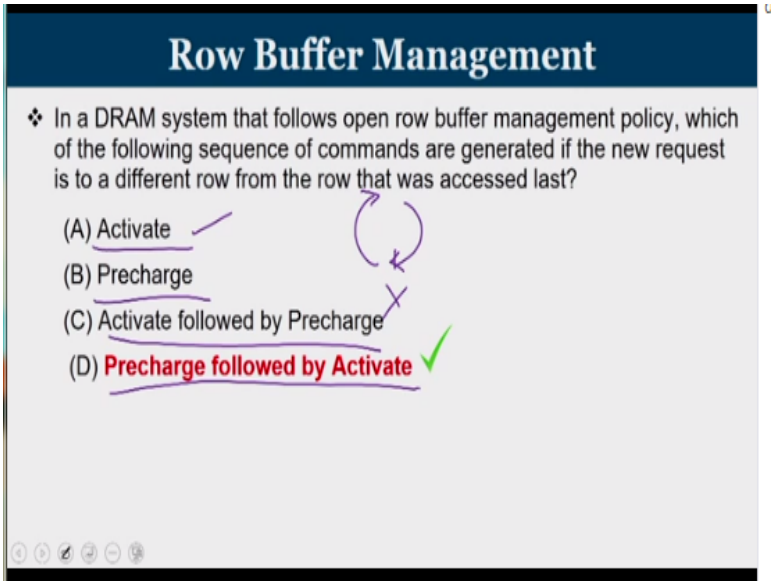


Similarly for the same question if the system uses a 128 byte last level cache and is using cache block interleaving in banking, which bank the physical address 0x2244668AA is mapped to. So here we bothered about 128 byte last level cache this concept also we learned in our lecture video. So we need to carry 128 bytes together because you have to fill up the last level cache, generally data from the DRAM is been copied to last level cache.

So this 4 + 2 that will give you 7 bits and that will give you 2 power 7, 128 bytes means you keep all these locations same address you vary the last 7 bit that will give you 128 bytes data to be transferred from the DRAM onto the way to the last level cache. So since you have total of 13 bits in the column out of it is only 4 bits are used to here, so your column bits get separated and the bank bit will come in between your column.

So why this separation happens here because I require total of 7 bits at the end, so adjacent cache blocks are mapped on to adjacent banks, so this is the 3 bit of your interest. So when you take this the address that is been given, these are the last 12 bits I write the last 12 bits out of which 7 is been extracted. The next 3 bit will tell me which is the bank, so we are working with bank number 1.

(Refer Slide Time: 11:54)



This question is about row buffer management, in a DRAM system that follows open row buffer management policy which of the following sequence of commands are generated if the new request is to a different row from the row that was accessed last. So we had to see it is a open row buffer management, so here we have a bank where anyhow rows and columns, you have the row buffer. Now we have a queue, this is in the DRAM controller we have the queue of request.

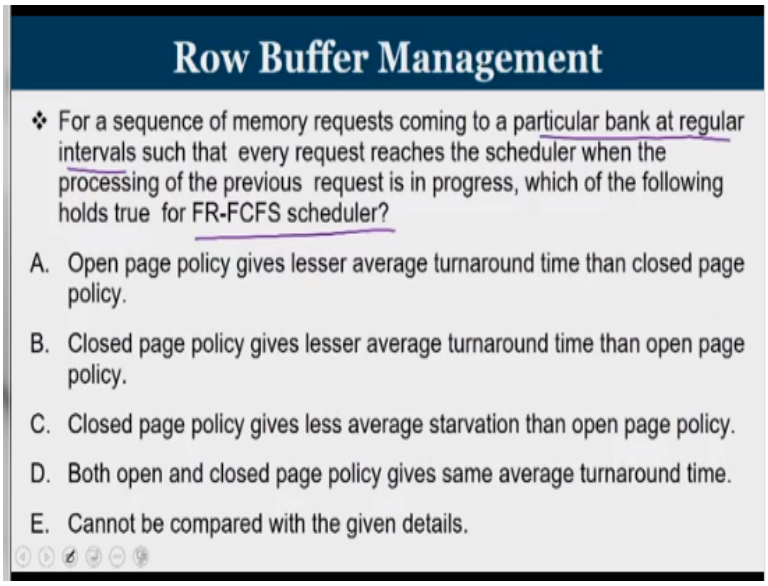
So in a DRAM system that follows open row buffer management policy, which of the following sequence of commands are generated if the new request is to a different row. So I got a new request this is totally different, let say this part of row number a and now we have row number b that is rare in the row buffer. So row buffer carries 1 row, the new incoming request is to a different row what we will do.

So in this case I have to store the contents of b back into the corresponding row there is one signal and that signal is known as precharge. And then you have to activate the new row let us say this is the row corresponding to a, so we have to bring it back, so it is activate. So in this context the answer is precharge followed by activate, so activate alone means is the contents of a particular row is transferred to row buffer.

Precharge means contents of row buffer is transferred back to a row. Activate followed by precharge that means you transfer something into a row and then you transfer something into the

row buffer and then you store it back from the row buffer into a row, that is not relevant here. So precharge is what is been to be done first followed by the activate signal.

(Refer Slide Time: 13: 35)



Next one is also a row buffer management question for a sequence of memory request coming to a particular bank at regular intervals such that every request reaches the scheduler when the processing of the previous request is in progress, which of the following holds true for FR-FCFS scheduling. So there are 4 statements given open page policy gives lesser average turnaround time than closed page policy, closed page policy gives lesser average turnaround time than open page policy.

Closed page policy gives less average starvation than open page policy both open page and closed page policy give same average turnaround time. We cannot compare open page and closed page in this context with whatever given details, so we have to understand what is the concept of turnaround time. So when you have a request that is coming into the DRAM queue that time onwards until the request is been completely service that means the content that is been asked by the request is been transferred to the DRAM controller that is called turnaround time.

So you have a bank, which has a row buffer and now some contents are existing there. Now we are in the process of extracting the data, so every new request is coming to the queue in such a way that prior to the completion of these previous operation that is been under service, new

elements are coming, meaning at the end of the operation there have some elements already in the queue. Now what is the difference between an open row policy and a closed row policy.

In the case of an open row policy, after servicing a request we keep the row open hoping that the future request will be to the same row. So if the future request is to a different row then you have to do a precharge and then activate the new row. If it is a closed row policy by the time you complete the servicing of a given request if the queue is empty then we do not know what is going to come in the future.

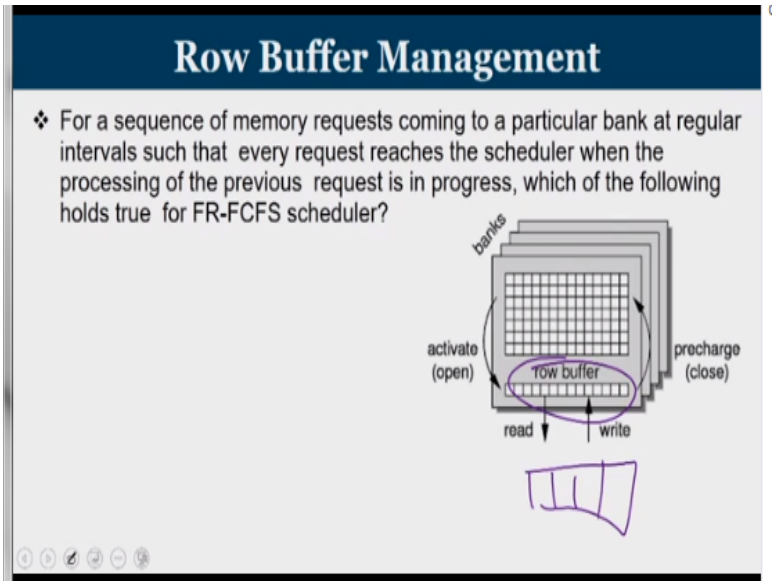
So you perform a precharge operation hoping that the future request that is going to come is to a totally different row. So in open row policy, we hope that the future request is to the same row leaving the row open, in closed row policy we hope that the future request will be to a different row. So there is no point in keeping the row open right now, let me close it that is called precharge. Now in this context in the given question, we have always your buffer occupied with some requests, there would not be any case of the buffer empty that is what is been told.

When the previous request is in progress or processing of the previous request is in progress, the new request is coming. So the new requests are always coming when the previous one is in progress means when the previous request is completed, we do not have an empty queue. If you do not have an empty queue then in the case of an open row policy we can search for FR-FCFS scheduling whether there is anything to the same row.

If so apply only cache if it is so different signal perform precharge and then activate, in the case of closed row also, as long as you are applying FR-FCFS scheduling first ready first come first serve, if at all you find anything row buffer hit try to pick up that request in the queue. If you are not able to find out anyway you have to activate a new one, so it is precharge. So both row buffer management policy and closed row buffer management as well as open row buffer management are going be picking up request in the same order.

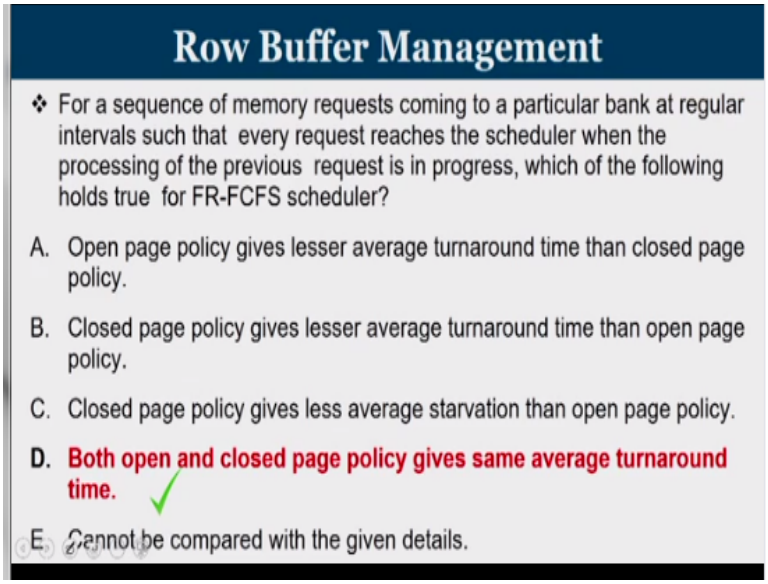
So there is no difference at all, so the answer is the turnaround time is going to be same in the case of both open row and closed row policy.

(Refer Slide Time: 17:11)



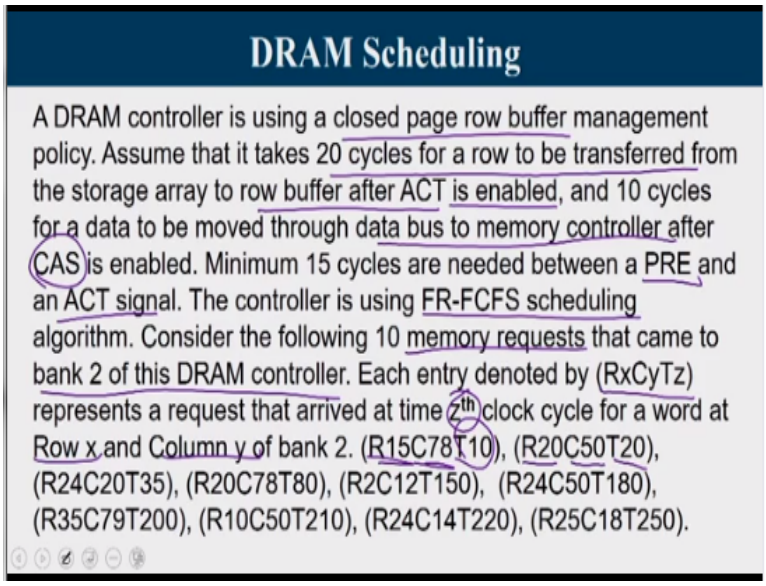
So this is what we have told, so we have always whenever we are progressing something, we have a queue and the queue is non empty as long as the queue is non empty open row policy and closed row policy would not make any change at all.

(Refer Slide Time: 17:24)



So both open and closed row policy will give same average turnaround time.

(Refer Slide Time: 17:28)

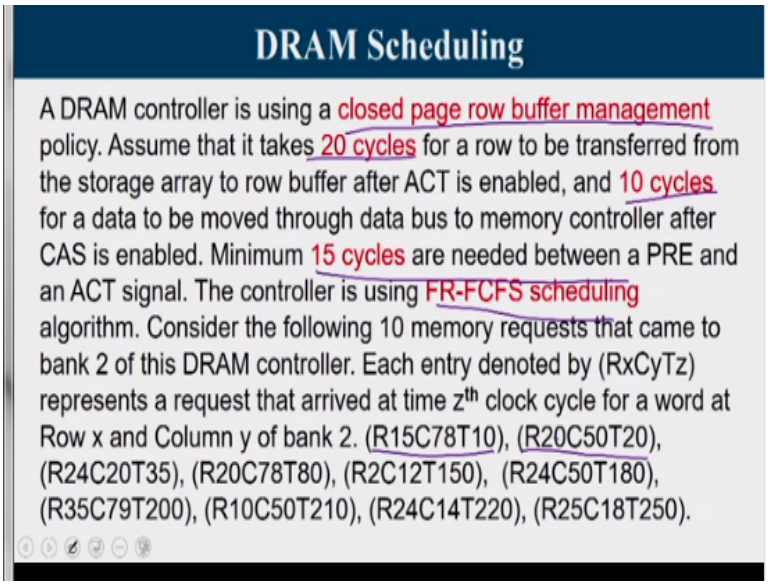


Let us now come to a scheduling question in DRAM, a DRAM controller is using a closed page row buffer management policy. Assume that it takes 20 cycles for a row to be transferred from the storage array to the row buffer after they activate signal is enable. So once you give an activate signal it take 20 second for the data to reach row buffer and then it take 10 cycle for the data to move through data bus to memory controller after the CAS- column address drop is enabled, minimum 15 cycles are needed between a precharge signal and an activate signal.

The controller is using FR-FCFS scheduling algorithm, consider the following 10 memory request that came to bank to of this DRAM controller. So you are going to get few requests, all of them are to same bank, so we are dealing with only one row buffer. Now the way in which the request are represented is RxCy and Tz which represents a request arrived at the Zth clock cycle, the last entries is a clock cycle for a word at row x and column y of bank 2.

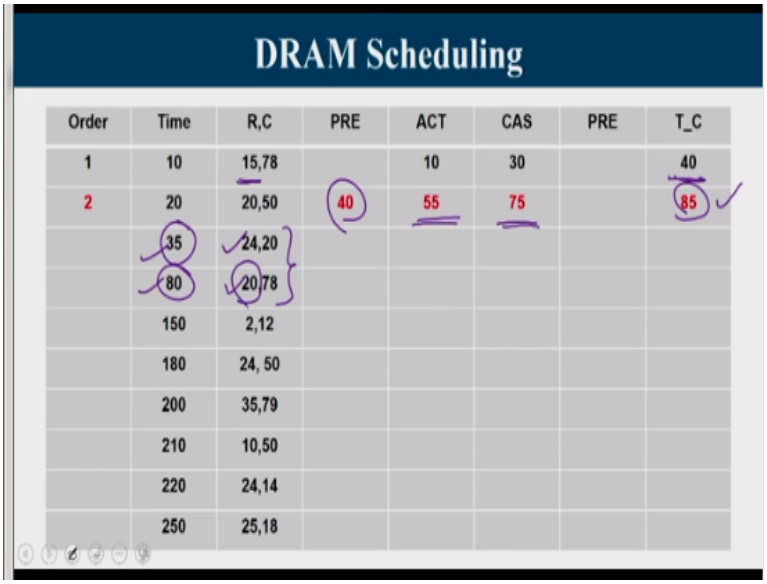
So this means row 15, column 78 the request came at time 10, row 20, column 50 at time 20, so you have a sequence of request. Now we are suppose to know what are the time in which the signals are been applied or by what time you can complete the operation.

(Refer Slide Time: 18:49)



So these are the important things from question, it is a closed row buffer management policy. So as long as so when you complete servicing a request if the queue is empty then we precharge that is the meaning of closed page row buffer management policy, 20 cycles between an act and the CAS, 10 cycles between CAS and transfer of data, 15 cycles between a pre and an act FR-FCFS scheduling algorithm.

(Refer Slide Time: 19:21)



So what is do is let us try to organize the given data whatever data is been given we are going to write it as a row number, column number followed by the time at which this arriving. So if you look at the very first data, first one is R15, C78 and T10, so R15, C78 and T10 time is 10. The

second one is 20, 50 and 20. So at time 20 that is a third parameter in row number 20 and column number 50 we have the data.

So I am organizing the data in the order of time of arrival and these are the rows and columns. Now let us try to understand initially the row buffer is empty, now when you look at time 1 there is nothing there in the queue and time 2 there is nothing. So the first element reaches only at time 10. As and when it reaches we have to schedule it, so we will give an activate signal because already the row buffer is empty.

So when you give an activate signal it take 20 cycles for the data to reach corresponding row buffer from the row. And then so it takes 20 more cycles, so at time cycle equal to 30 only, the data is ready in the row buffer then you apply a CAS signal it will take another 10 more cycles. So the transaction is complete at clock cycle 40. Now when clock cycle 40 if you look at these are the 2 request that have already reached the scheduler.

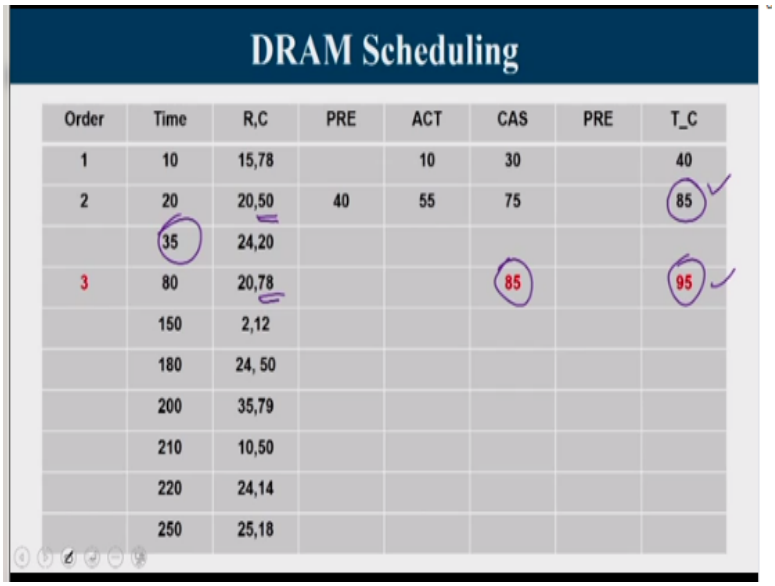
So if you look at that one is to row 20 other one is to row 24, so my scheduling is FR-FCFS, the current row that is opened is 15. So there is no other request that is there in the queue which is to the same row, so the ready concept is not there FR-FCFS first ready, no other row is ready. So we have to go for FC-FS policy, so out of these 2 request it is this request which came first. So I am going to do that how will I do, so at 40 my previous one got over.

So at 40 I am performing a precharge by which row number 15 which was there in the row buffer is written back to the corresponding row, at 48 generate a precharge signal it will take 15 cycles to complete the precharge operation making the time by 55. Then I have an activate it will take another 20 more cycles. So at 75 I can give CAS 10 more cycles, so at 85 I will complete. So at 85 if you look into these are the 2 elements that are already there in the scheduler queue.

But we see that one is to row 24 and the other one is to row 20, so even though the request which came late at clock cycle 80, it is to row to 20 and column 78 whereas the request which came at clock cycle 35 is to row 24, currently the time is 85. So these are the 2 elements from which one I will pick the, first one, even though it reaches first it is to a different row, the second one it is to

the same row. So as per our policy, whenever we are able to service a request that is to the same row, we have to do it first that is called FR-FCFS.

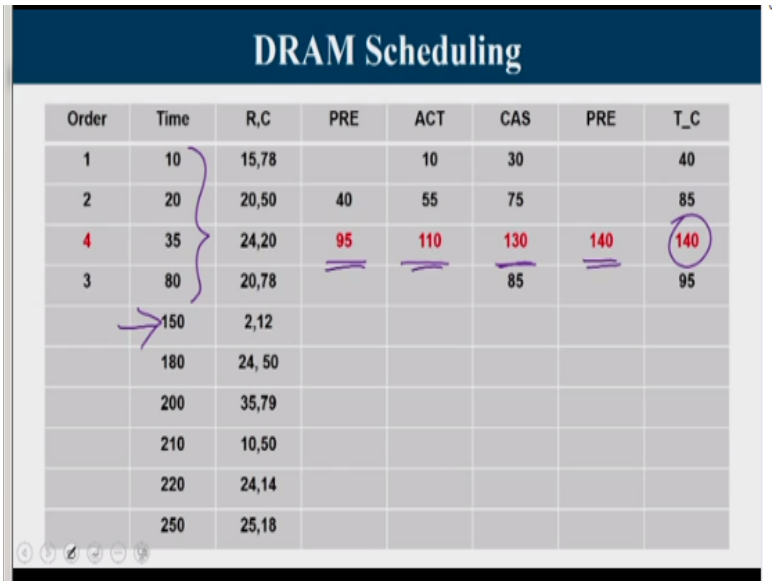
(Refer Slide Time: 22:23)



So now rather than the third one, it is the fourth one that gets scheduled first, so here we have to apply only a CAS signal, at 85 we complete the previous one 85 you can apply the new column why it is new column means previously I serviced 50 now the request is to column number 78. So just give only column number and another 10 cycles you are completing. Now the peculiarities at clock cycle 95 we are completing the transaction with respect to the third request.

Now we have to go to the new one, so at 95 at clock cycle 95 which is the one that is pending only this is pending, so there is no other way we have to precharge it.

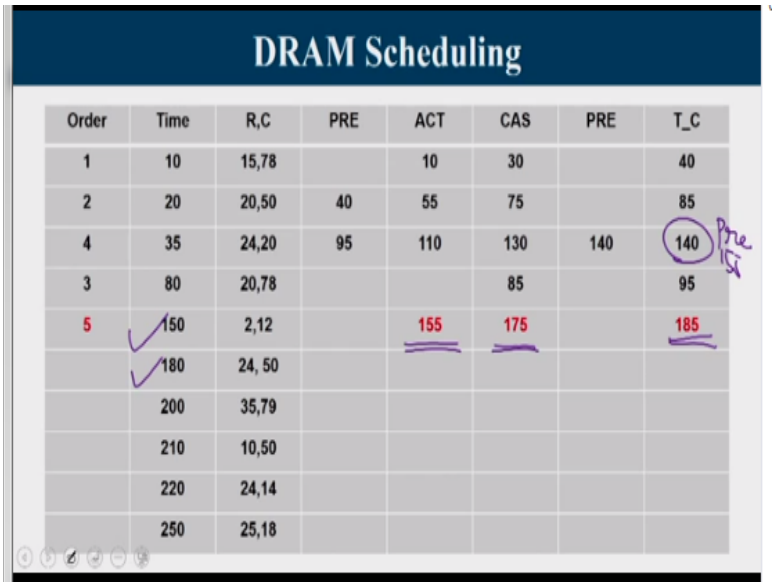
(Refer Slide Time: 23:04)



So that is the fourth one we precharge at 95 it takes 15 cycles, so at 110 precharging is over generate the activate signal, we take 20 cycles more at 130 activate is complete and the data is ready in the row buffer. We generate CAS signal at 130 by 140 it is over and then the transaction is complete at clock cycle 140, now what 140 we do not have any other request, the new request is coming only at 150.

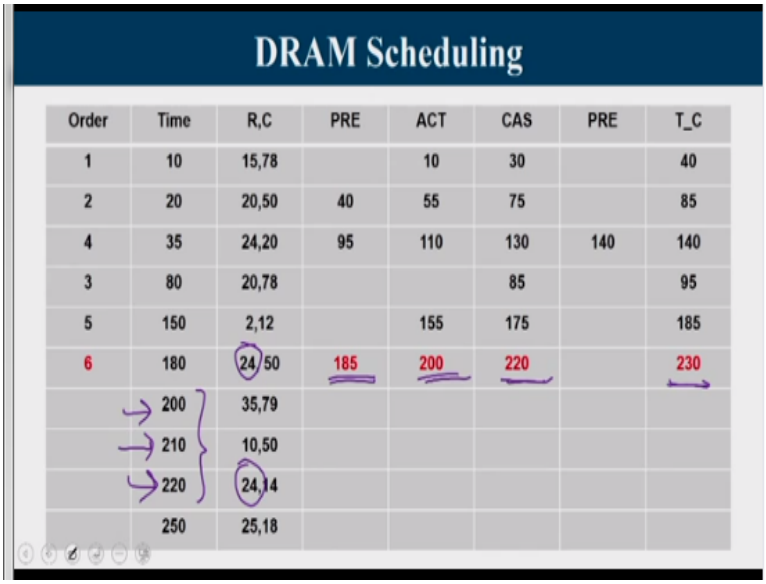
So at 140 we do not have any other request that is been there is a closed row policy, so we hope that in future whatever request that is coming is, so different row.

(Refer Slide Time: 23:43)



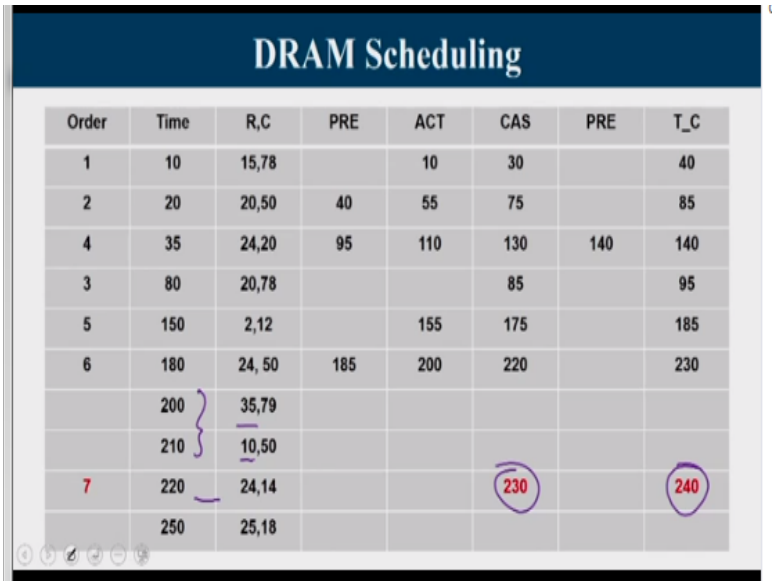
So at 140, what I do is I am doing a precharge operation, which will be taking time up to 155, so even though we have a request at 150 we can start the activation only at 155. Because at 140 we have given a request for a precharge because it is a closed row policy and it get over at 155. So at 155 act is given 20 cycles, 155 + 20, 175 another 10 more for CAS, so by 185 the transaction is complete. If you look at 185 what do you have left, you have only one request that is been left and that is to a different row.

(Refer Slide Time: 24:24)



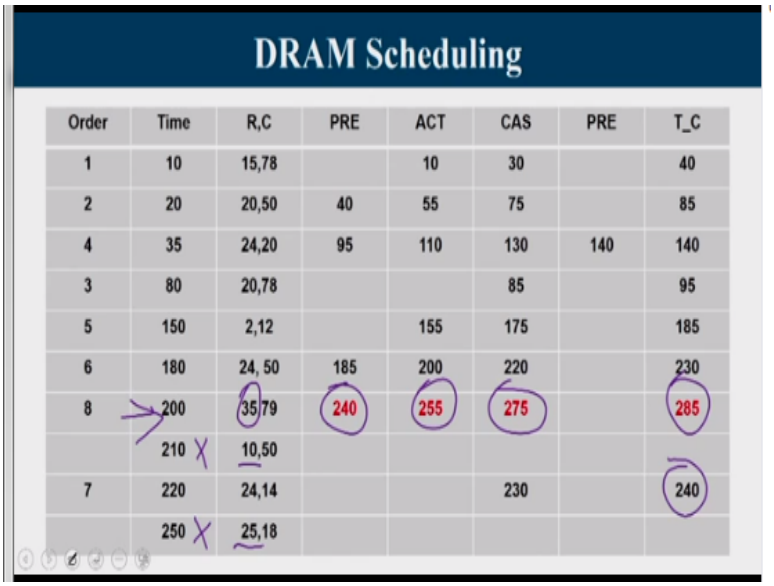
So there is no other way at 185 you are giving precharge and then you are giving an activate at 200, 220 CAS and 230 the operation is over, by 230 these 3 requests are already there. So you have to understand that this is now row number 24, if you look at the remaining 3 request one which came at 200, other which came at 210 and the third one which came as 220 this one is to the same row. So as per the policy I will service the one that is having a hit.

(Refer Slide Time: 24:55)



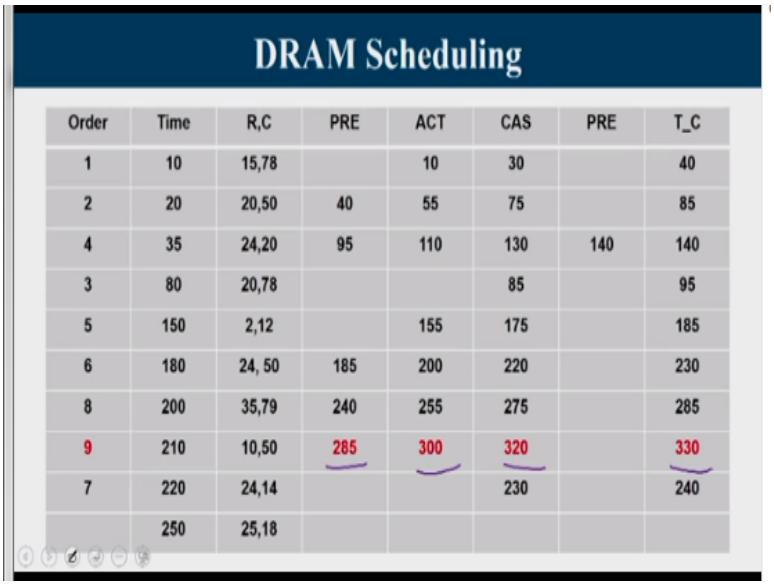
So the seventh one that is been service this one, so we have to apply only CAS at 230 240 it gets over. Now by 240 these are the 2 ones that are already there, out of which none of them is to same row number 24, so you go as per FC-FS policy.

(Refer Slide Time: 25:13)



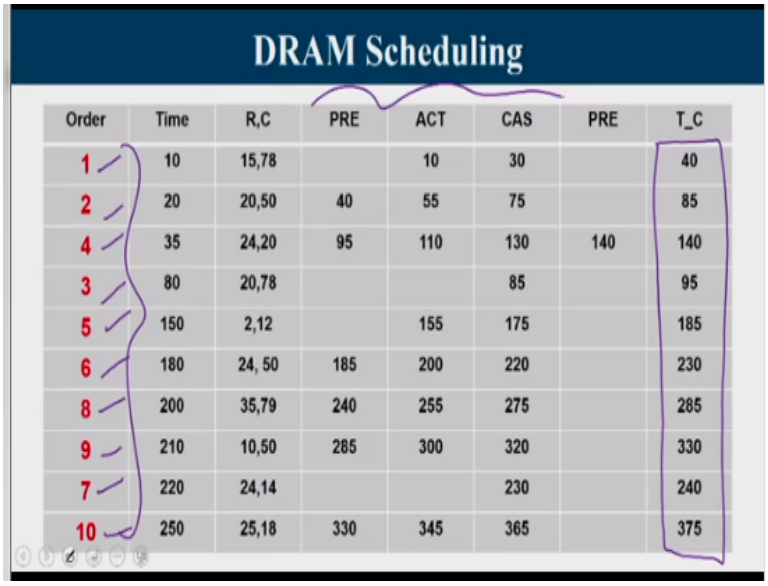
So that makes this request that is to be serviced at 240 you give precharge, so you are completing a transaction at 240 the same cycle precharge is been given, 15 cycles activate is given, 20 cycles CAS is given, 10 cycles at 285 you complete. So by 285 what we have is we have these 2 elements also ready, now currently I am processing row number 35 none of them is to same row, so you give which one came first.

(Refer Slide Time: 25:41)



So one which came at 210 that is been serviced with the timing 285 for precharge clock cycle 300 for activate, 320 for CAS and 330 the transaction is over.

(Refer Slide Time: 25:50)



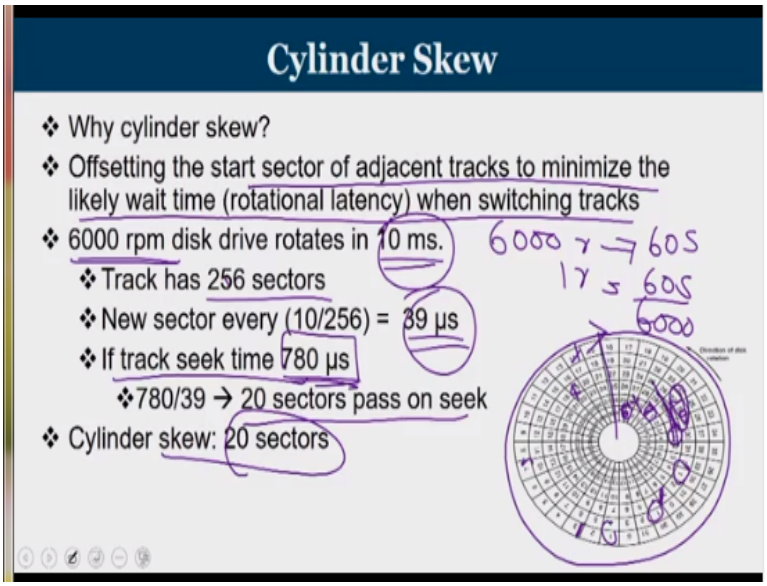
And then you have the last one that is getting serviced, so the whole thing will get over at clock cycle 375. So now it is been asked what is the order in which it is been serviced 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10 this is the order in which the requests are getting serviced. And these are the clock cycle in which the DRAM controller receives the data and these are the times at which precharge act and CAS everything is been done.

So in this question, what is given is we are been given a set of request that is been coming to a particular bank of the DRAM controller. And now we are trying to find out what are the order or what are the timings at which various signals are been given. So as far as DRAM is concerned, the time at which the signals are given is very important. So as mentioned in the question, after you give an activate signal it takes 20 cycles for the data to reach the row buffer from the corresponding row.

So the CAS signal can be applied only after 20 cycles, the circuitry of the DRAM has to be designed in such a way that these timing constraints are to be strictly followed. So this gives us a different picture of how even though the requests are coming in an order, the way by which they are been serviced is based upon the scheduling algorithm. So FR-FCFS is the most common scheduling algorithm that is been followed.

When you are done with the request look into the queue what are the set of requests that is been available and pick up that request, which is going to have an exact row match. The row number of there incoming request is same as the row that is been currently kept open in the row buffer. If you are not able to get a match like that, then you go as per FC-FS scheduling, this is the way in which it means to be done. I request you to practice few more problems in this to get more clarity as far as the timing and the signal generation of DRAM controller is concerned.

(Refer Slide Time: 27:44)



Now let us move onto the hard disk, this is a question with respect to cylinder skew, if a 6000 RPM disk has 256 sectors per track and the track seek time is 780 microsecond, what is cylinder skew. So what is basically cylinder skew, offsetting the start sector of an adjacent track to minimize the likely waiting time when you switch on to the tracks. So this is the cylinder let us say you can see that here we have one that this sector number 1 of a track.

But if you look into the sector number 1 of the next track is not exactly in the same and it has been shifted, sector number 1 of the next track is shifted, sector number 1 of the very next track like that if you look into the sector number 1 of various tracks are not exactly in the same line they are being slightly positioned, what happens is when your head is now currently touching sector number 1 and go all the way read the data up to sector number 31.

Now you want to take the data from the very next cylinder, so the head is going to get itself detached from track 0. Let us say here and then it is moving to the next track, by this time since the disk is rotating by the time you take off from track number 0 and come back and touchdown track number 1, some of the sectors have already slipped. So to avoid that sector number 1 or 0 whatever is beginning of the sector of the next track has to be slightly adjusted or shifted.

Such that during this takeoff and landing time of the head, you are not going to lose your data. So it is actually a 6000 RPM disk that means the disk is going to rotate at 6000 times it is going to revolve around the axis in 1 minute. So 6000 revolutions you are going to make in 60 seconds, that means 1 revolution will take 60 divided by 6000 that is going to be 10 millisecond, 60 seconds divided by 6000, so it takes 10 millisecond to complete 1 rotation.

1 revolution = 60/6000 = 10 miliseconds

Now it is been mentioned that you have 256 sectors, so completing 1 will take 10 millisecond, so during this 10 millisecond you are moving across 256 sectors. So how much time you are going to spend on one sector roughly 39 microsecond you are going to spend on a sector. So if it requires 10 millisecond to cover up 1 complete track that means 256 sectors approximately 39 microsecond is what you spend, so 10 millisecond divided by 256 it is 39 microseconds.

Time for 1 sectore= 10ms/256 = 39 microsec (approx)

Now it is been given in the question that track seek time is 780 microseconds, so that is a we were going to take off from one of the track and then going to touch down the very next track. So during this time, so that is a time given track seek time it takes 780 microseconds to switch off from 1 track and touch the adjacent track. So during this 780 microseconds time, you actually pass through 20 sectors, so 20 sectors have already gone by the time you touch down.

So in the adjacent cylinder, the next sector has to be skewed or shifted by 20 sectors that means here it is 1, the one of the next track should be kept at a distance of 20 sector. So somewhere here than here like that, it is a way how cylinder skew is been calculated. So computation of cylinder skew will help us in finding out how much should be the shifting that is been done in adjacent cylinders as far as numbering of the sector is been concerned.

Then there is yet another important section as far as disk is concerned that is about scheduling given a series of request, you have to find out what is the order in which they have schedule. We have learned different scheduling algorithm like FC-FS, SSTF shortest seek time first scan algorithm, look algorithm then the elevator algorithm like that. Already during the lecture session, we have taken 1 example and found out what is the average turnaround time and waiting time of all these.

So when you get a sequence of request, the request can be of the form platter number, track number, sector number and all. But as far as scheduling is concerned what we want is only track number which is eventually the cylinder number. So scheduling is done with the help of cylinder numbers, so from the given set of request extract the cylinder numbers alone and then based on the cylinder number apply the algorithm whatever been asked.

So since already we have discussed with the scheduling algorithms with numerical problems itself during the lectures, there is no specific question that is been planned as far as tutorial is concerned but kindly workout maximum number of questions. So by this we have almost come to the penultimate week of this course. I hope you are now preparing for the final exam as well kindly do go through all the videos that is been given.

And this tutorial sessions where numerical problems are worked out, which will give you more grip on the topic. And kindly do prepare well for the final exam before just one more week that is been left thank you.