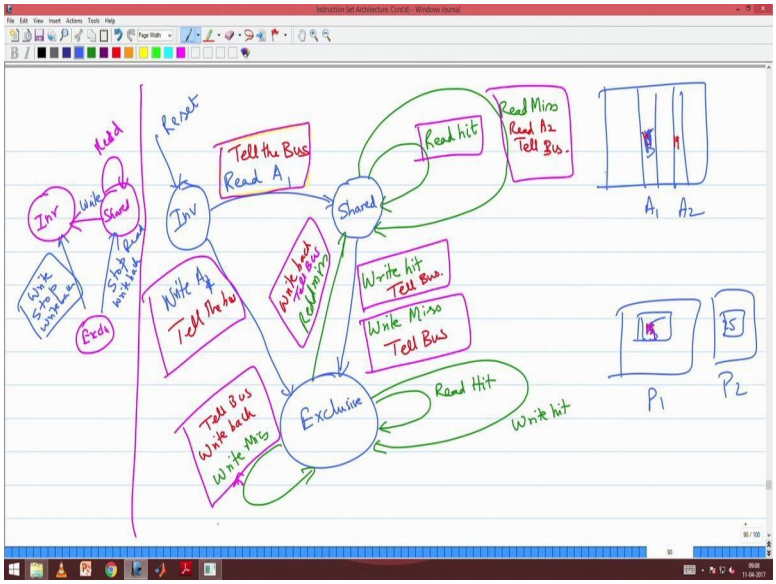
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Lecture – 39
Shared Memory Architecture, Cache Coherence

So last class we solve this cache this is for the multi processor cache. And so, each cache line as a state. We are only one state in our earlier uniprocessor cache, but we have 2 bits meaning one bit for the state now we have 2 bits.

(Refer Slide Time: 00:43)



And we need 2 bits because we need to have we need to maintain 3 states, is just not invalid and valid, but it is invalid shared excusive. And the state of cache line will change because it is own processor does something which is this larger state machine that you see here, this is this machine that you see, while it can also be changed because of some activity on the bus, which is seen shown by this smaller machine.

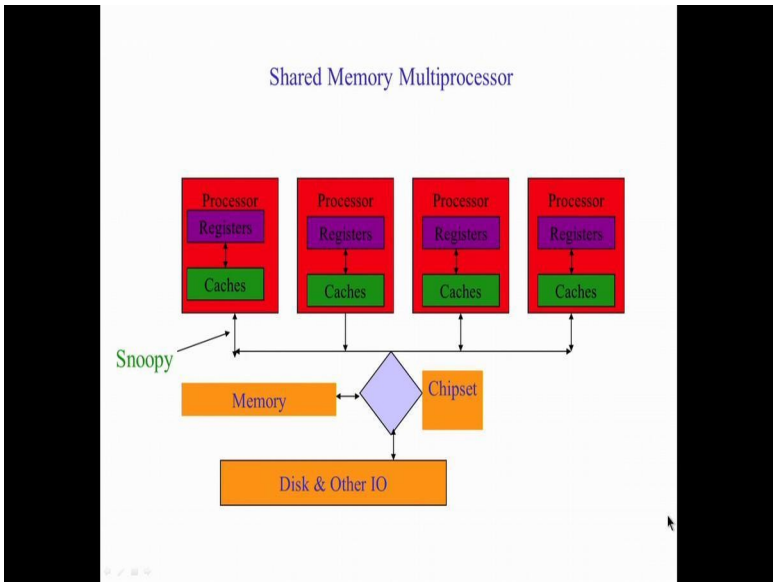
So, so there is a simplest of the protocol it is a token ring protocol, where in there is a common shared bus and each processor gets that token that is one token that is rotating.

When I have the token as a processor I become the master of the bus and I can do read and write on the memory. And that is what happens because of the read and write to my own cache is given on the right, hand side the larger one, and when I do not have the bus I do not have the token I am not the master of the bus, other point of time I will be

snooping on what is happening and that is given by the left state machine. And when something is happening here there is on the larger state machine when something is happening because of my CPU, I keep broadcasting some massages on the bus So, that I can basically take care of the cache coherence issue right.

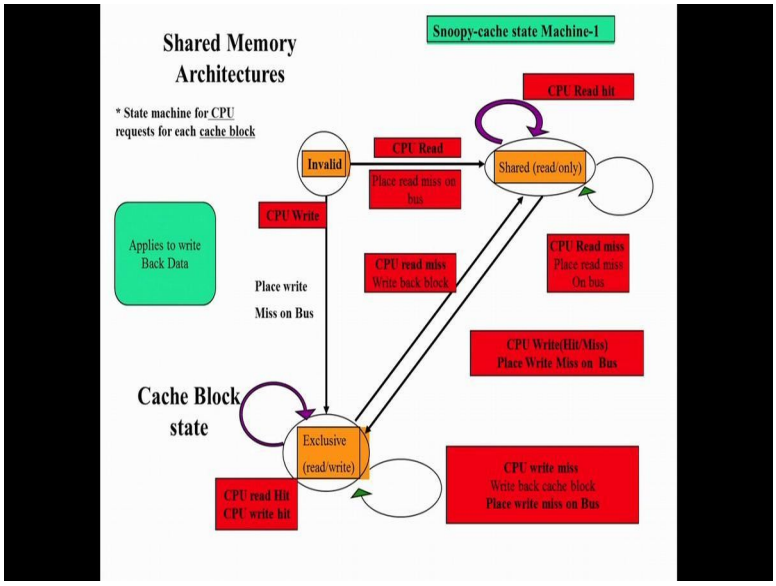
You all get this? So, this is this is what we saw in the last class. And now we will do one example and then go into the what we call as the parallel models of computation. So, let me ok.

(Refer Slide Time: 02:21)



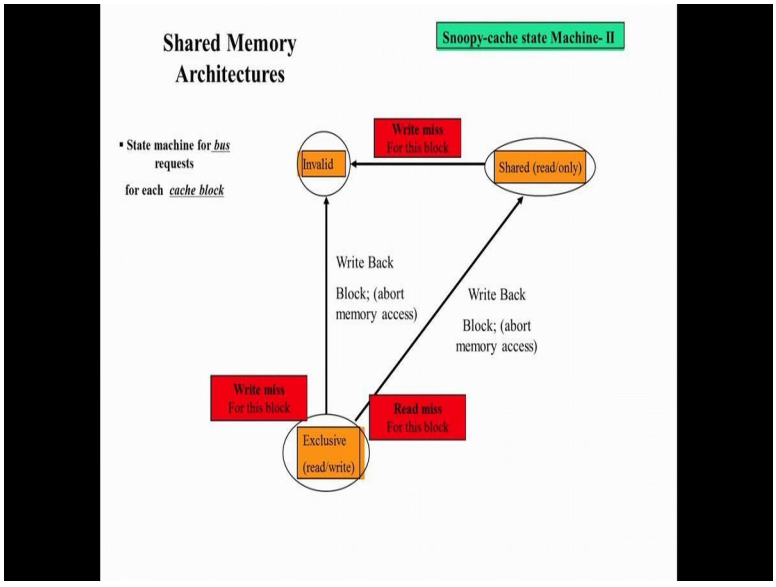
So, this is the model of our system. We have a processors, each process has it is own individual register there are caches. And we are following a write back policy and there is the memory is connected to the common bus. And that bus we call as snoopy bus because it allows you to snoop.

(Refer Slide Time: 02:43)



And now this is one state machine the largest state machine, this is the state machine for CPU request there is the snoopy cache state machine 1. This is something because of my processor, this is what happens to your state of a particular cache line based on it is own processor trying to read or write from the memory.

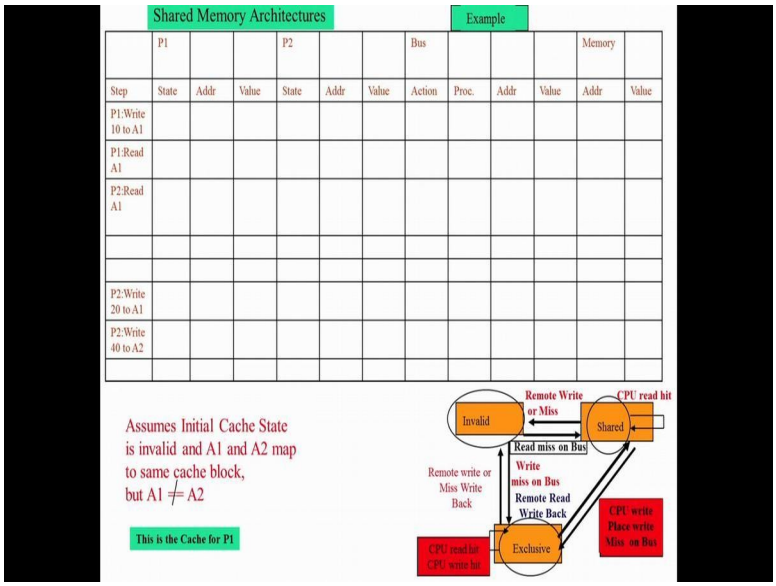
(Refer Slide Time: 03:06)



And this is the smallest state machine. So, whatever I explain is there in these 2 slides ok.

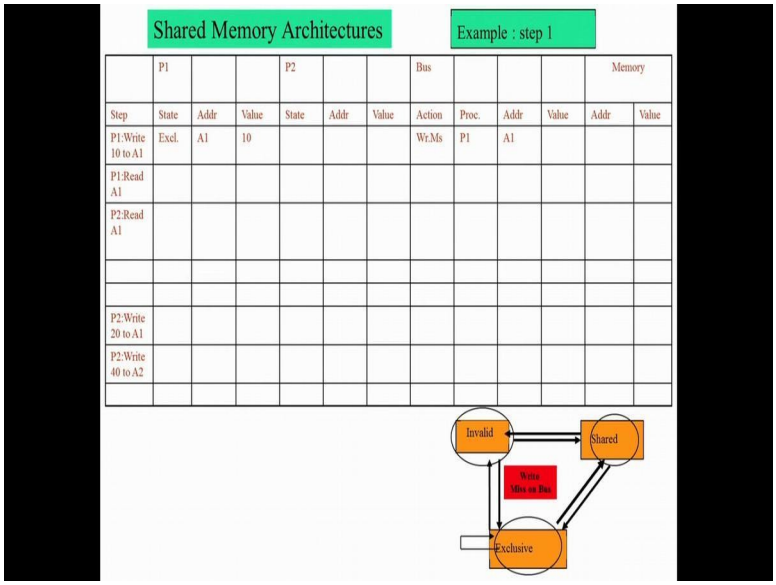
Now, let us talk let us start doing this. So, this is one example we have 2 addresses A 1 and A 2. A 1 and A 2 map on to the same location.

(Refer Slide Time: 03:12)



And let us say it is a directly map cache. So, either A 1 will stay or A 2 will stay. Now this A 1 and A 2 are shared by say 2 processes P 1 and P 2 right, now these are all the actions that ah you know P 1 does, P 1 writes the value 10 to A 1 as you see on the left move side P 1 write. The value 10 to A 1 the P 1 reads A 1 then P 2 reads A 1. At the same time P 2 writes 20 to A 1 P 2 writes 40 to A 2 right, now P 1 and P 2 are 2 process course and they have the own I 1 cache. This is A 1 and A 2 are addresses in the memory which will map on to the same location in that cache. So, and let us assume for our example that it is a directly map cache. So, either A 1 can stay or A 2 can stay. Now what will happen? So, this is one small case study that we will do to just you know re explain that state machine. I hope you are understood what the state machine is.

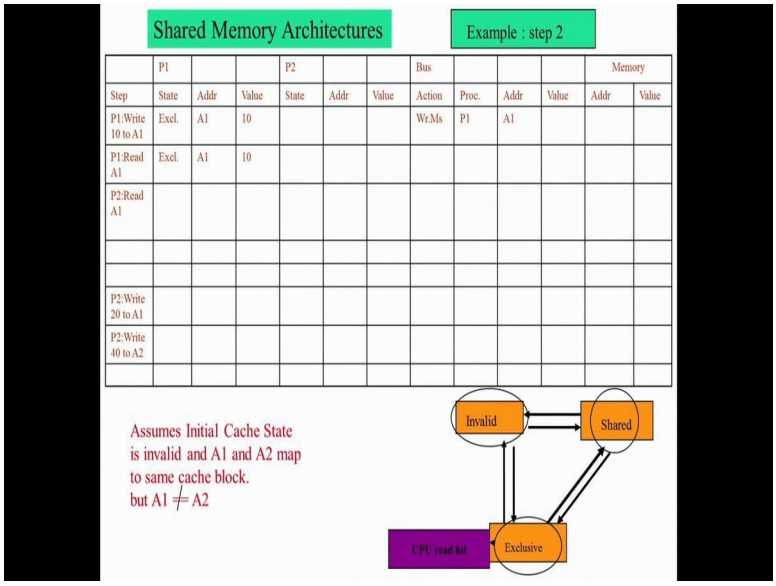
(Refer Slide Time: 04:42)



So, initially the caches are all invalid. So, the line in which A 1 or A 2 is getting stored in both the caches. Both the I one caches there in invalid state. So, so this is state of. So, we start of here. So, what happens? So, first one P 1 writes P 1 writes the value A 1 to ten. So, in the P 1s cache A 1 will be stored and it will have exclusion right. So, from invalid when there is a write miss it goes to exclusive. This particular link please note here this link is exercised. So, the value of A 1 in the memory and the value of A 1 in the I 1 cache of P 1 are different. So, right. So, I have written the value 10 it can be anything there probably they can be different. So, I have A 1 as an exclusive address. So, P 1 as A 1 as a exclusive address in it is own cache right. And this is this particular line please see the bottom state machine I have marked it green. So, this is that particular line getting are you able to follow? Right, now So, what will happen? On the bus I will say there is a write miss on A 1. I need to tell because there could be somebody who is having the value of A 1 in exclusive shared state and now I have to say that now I am the exclusive owner of A 1.

So, I will on the bus I will announce that there is a write miss, who will announce? P 1 will announce that there is a write miss of address A 1, nothing actually happens to. So, memory the value of A 1 would be something some junk we do not know right, are you able to follow this particular steps, any doubts? Next now P 1 reads from A 1 and P 1 reads from A 1 it is already in exclusive state it need not tell anyone right.

(Refer Slide Time: 06:25)



It is read it just read it is just a read it, nothing happens on the bus. Please note that this is very, very important, nothing happens on the bus is empty. Because I am just reading A 1 and this is that loop here this is a read hit on that loop here. So, here itself there was a read hit. When there is a read hit I did not tell anyone because, nobody is going to benefit by knowing that I am reading A 1, because I am holding A 1 in the exclusive state. Are you able to follow? Yes or no? Ok.

So, why should I not keep on broadcasting on A 1? Now on the bus if I keep broadcasting on the bus then there is there is going to be more power consumption right, when I send something on the bus everybody will snoop every cache controller will see what is happening. And there will some energy spender. There will be some power consumption there. So, more I optimised on sending massages on the bus please understand the more that I am going to basically benefit in terms of power consumption right.

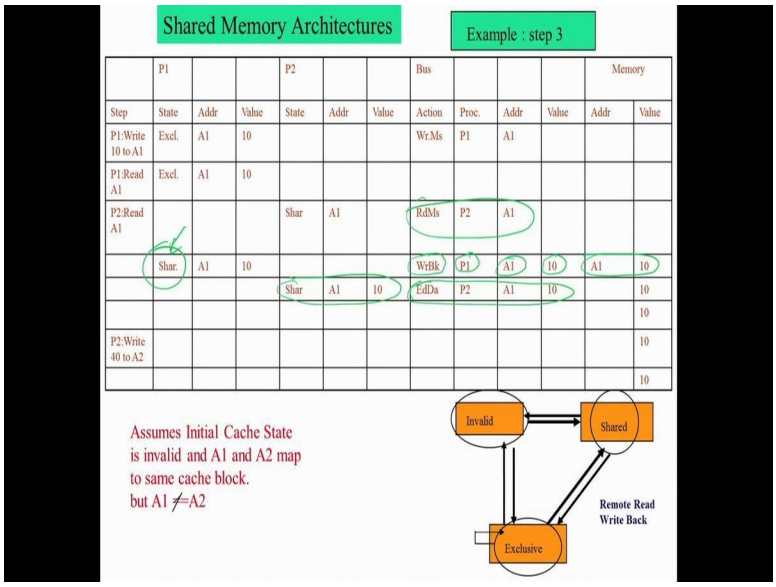
So, in this case there is no necessity for me to tell if I holding A 1 in exclusive mode by this state machine there is no necessity and if there is a read hit for that A 1 itself there is no necessity for me to go and tell everybody else a I am reading A 1. Normally today when you take modern architecture in this type of a token architecture there will be 8 course upto 8 course they have scaled. So, the remaining 7 course if I just put something on the bus the remaining 7 course will start processing. What they should do? They will

take that address A 1 then they will say whether it is stored there and if the valid bits there. So, they have to do all this check.

So, when they see something on the bus it essentially means they are going to access the cache to see whether that address is stored there correct right. So, unnecessarily if I send a message then some other 7 fellows will do a cache accesses is not that this cache controller snooping that value it take that value then go to the cache and ask a is there something related to this, if there is something related to this then it as to take action, if I am storing the address A 1 and something happening to address A 1. Then I have to go and check what should I do with that.

So that means, the movement some massage is going on the bus my cache gets accessed. Please understand the most important thing. So, this is where we start gaining on the power see why suddenly we have you know, what do you mean by optimising on power? These are all the small things right, these are all small intuitive step that we need to take care So that we optimise some power. Now let us see (Refer Time: 09:30) third step, I will go to the next step.

(Refer Slide Time: 09:41)



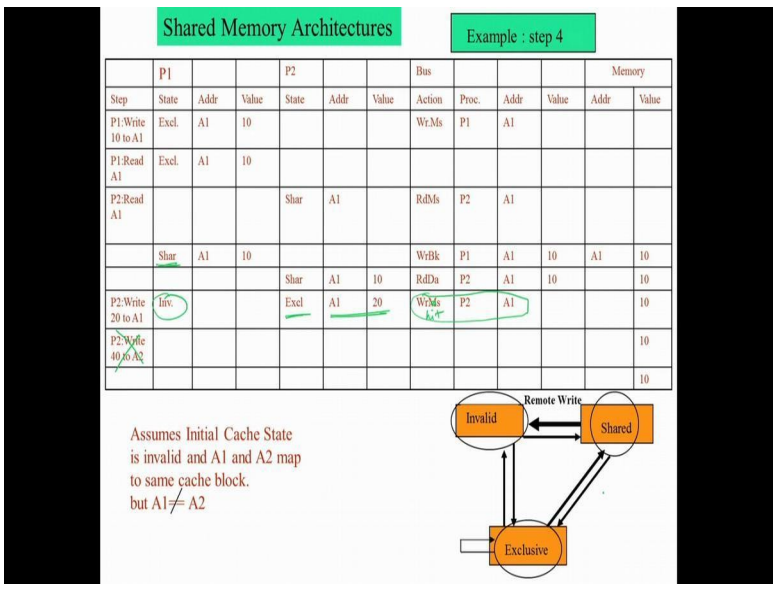
Now, P 2 tries read A 1. When P 2 tries to read A 1 what will happen? It so, let us be very careful let us note it here. It first finds a read miss for A 1, read miss it will it will it will it is trying to read A 1 it is in invalid state. So, it puts on the bus

that I am having a read miss and I am trying to write A 1 correct.

Now, what will happen? Your fellow immediately on the bus P 1 will say a stop I have A 1 in exclusive state wait. So, it will write back A 1, it will write back the value 10 on to the memory. So, on the address A 1 10 will be loaded, please note here P 2 now wants to read A 1, P 1 is having it in an exclusive state, right? Immediately what will happen? P 2 says that there is a read miss, on the bus correct? That by state machine we have seen right, from the invalid state when I have a read miss I will say I have a read miss on the bus immediately P 1 will be snooping because P 2 is the master right, P 2 is reading right, when P 2 wants to read P 2 will be the master right.

So, P 1 will be what will be spooning and now he finds A 1 now he finds A 1 in exclusive state in it is own cache. So, immediately it will say wait write back what the value 10 into A 1. So, when it right, back immediately in the memory you will see that A 1 gets the value 10 then what will happen? And then at the same time P 1 will also make it is state as shared state please note. Here because some P 2 will also have the value of A 1. Then what will happen? P 2 will read the data will read the data from memory P 2 will read the data and it will also make it to shared the state. Are you able to follow this? Right So now, when P 1 had address A 1 as an exclusive address, when P 2 wanted to read P 1 stopped that read meaning not stop suspended that read it wrote back the latest value of 10 into A 1, then it made itself it made it is state as shared and P 1 also P 2 also get seat in a shared state right, this is how the cache coherence problem is solved, is it ok?

(Refer Slide Time: 12:42)

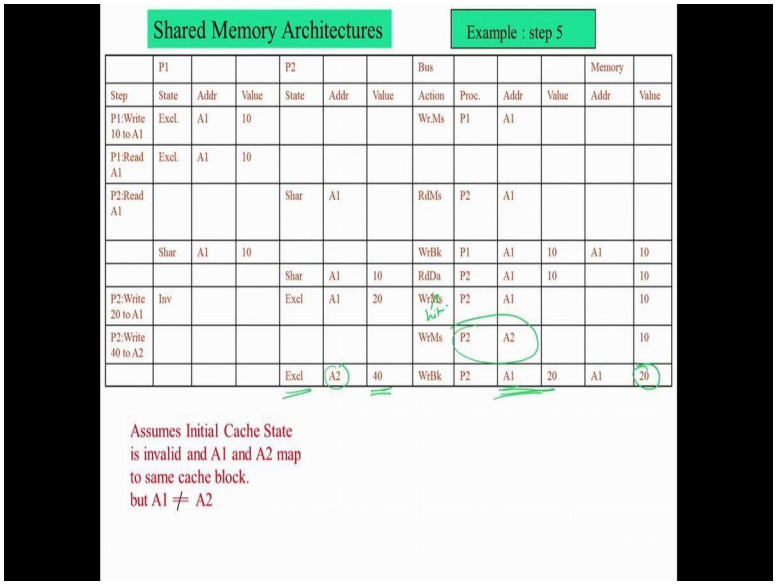


Now, what happens? The next step P 2 P 2 writes 40 to A 2. Is an next state. What will happen when P 2 writes 40 to A 2? P 2 writes 40 to A 2 what will happen? One minute sorry, I think this should be removed. Yeah, now P 2 writes 20 to A 1, forget this for a minute, forget this for a minute. P 2 writes 20 to A 1, now what will happen? Now P 2 actually should hold A 1 in exclusive state. Now P A 1 is in which state A 1 is in shared state both in P 1 and P 2.

So, what will P 2 do? When P 2 writes 20 to A 1 it will put a write miss on oh sorry, it should put a write hit, will put a write hit P 2 A 1, because A 1 is there in the cache. So, it will say I am writing into that cache, but that it will announce on the bus. The movement it is announce as and since it is writing it will it will make P 2 will make it as exclusive and A 1 is the value 20, while P 1 will be seeing the bus. Now it see that P 2 is writing into A 1 it was it is holding A 1 in shared state. So, it will make that thing as invalid, are you getting this? Yes or no? Are you getting this? Ok.

So, P 2 now when it want to write 20 into A 1, P 1 has A 1 in shared state it will make it invalid, while P 2 will make it state as A 1 state as exclusive. And then what happens there is a write hit? It also tells on the bus when P 2 wants to write 20 to A 1 it will say I am writing something to A 1. So, that the other fellows will give. So, this is from shared it to exclusive when I have a write hit I say on the bus that there is a write hit. So, that other fellows who are sharing can make it exclusive. So, the most important thing that you want we have to understand here is what is happening on the bus. Sometimes I announce on the bus sometimes I do not announce on the bus why I announce why I do not announce this example is trying to make it clear right.

(Refer Slide Time: 15:27)



	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	Value
P1:Write 10 to A1	Excl.	A1	10				Wr.Ms	P1	A1			
P1:Read A1	Excl.	A1	10									
P2:Read A1				Shar	A1		RdMs	P2	A1			
	Shar	A1	10				WrBk	P1	A1	10	A1	10
				Shar	A1	10	RdDa	P2	A1	10		10
P2:Write 20 to A1	Inv			Excl	A1	20	Wr.Ms	P2	A1			10
P2:Write 40 to A2							WrMs	P2	A2			10
				Excl	A2	40	WrBk	P2	A1	20	A1	20

Please note what would be the value of A 1, in the main memory it will still be 10, because this is a write back policy.

Now, in the next step I am writing 40 to A 2, I am writing 40 to A 2. Then what happens? There is a write miss on A 2. There is a write miss on there is a write miss on A 2. So, what will happen? And but A 1 and A 2 are sharing a location. So, I am trying to fetch A

2. So, I have to replace that A 1. So, what I do? I write back 20 back to A 1 and then store A 2 in the cache with value 40 with exclusive right, in the memory we do not know what is A 2 the latest value of A 2 is 40 which is in exclusive the value of 20 is basically written back into A 1, correct? Are you able to follow what I am saying? Yeah.

Student: should be write hit rate the first step.

40 to this should be write I thing I change it (Refer Time: 16:44).

Student: P 1 writes 10 to A 1.

That is a right, (Refer Time: 16:51) A 1 was not there A 1 was not there initially I am bringing A 1 right, initially both caches are invalid.

Student: (Refer Time: 17:01) compulsory.

Compulsory, this is write hit. This is write hit yeah.

Student: sir before P 2 writes 20 to A 1 if it write 40 to A 2.

No, it cannot because it 20 is in the cache right, from cache I should deposit in the memory and then only go and replace the cache.

Student: So, like first we should do write back?

First we should do write back and then write 40 right. Now did you all follow this example? So, what we have done is we have taken this state machine and we have basically, we have taken the state machine and basically we have explain the different step of this state machine. So, before we go to the So, essentially this is the state machine and we want to verify each of these actions, right? Now what we have done is by using the 4 or 5 cases we have verified some of these edges right, there are 1 2 3 4 5 6 7 8 9 edges. Every edge we should verify right.

So, suppose we have coded this state machine in an hardware description language like hack, you did hack right. So, suppose we have coded this machine we need to see whether it is working correctly or not. This is basically called functional verification. What we do we give test cases to verify if it is working correctly. So, what it means to it verify a state machine? First thing is we have to verify every edge. Then we have to verify every path from invalid shared exclusive, invalid excusive shared invalid shared in exclusive invalid. So, different parts I can take.

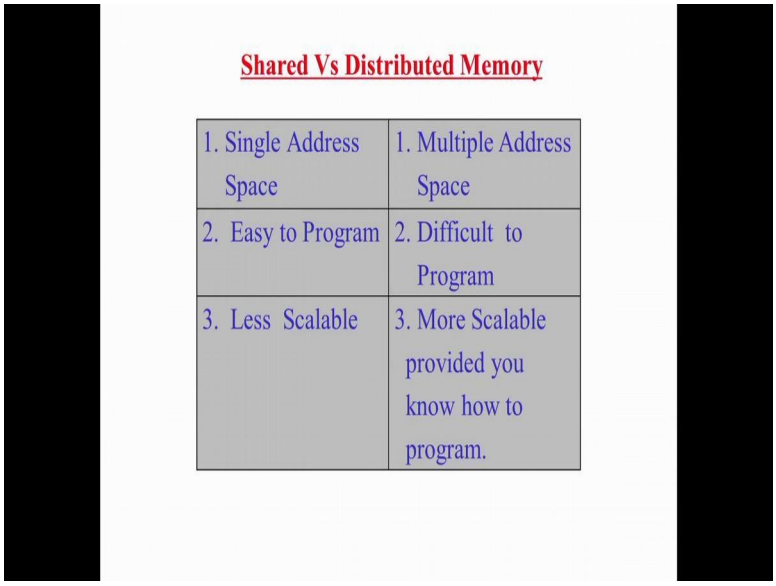
So, I that So, this is called one thing is called state coverage, state coverage means I visit every state right, that we have done. So, we have seen invalid shared and exclusive on both the state machines right, then there is something called edge coverage we have not done full edge coverage, but you done all most right, every edge we need to cover one cover one then there is path coverage. Every path we will like to cover right, there are infinite number of paths the movement I have a loop and then I could have infinite number of paths again right. But I need to come out with some comprehensive coverage. So, I need to see one of the important aspects of computer architecture is the movement I design a system I need to test verify that system whether it is working according to the specification. So, this is a specification and you will write the code in some very log or blue (Refer Time: 20:06) or some language or any system very log etcetera.

And now we need to see that whether that implementation essentially needs this specification. So, what are different types of coverage? Again I repeat state coverage edge coverage path coverage. We have done some part of it as a example. So, there is one famous question right, we are now talking of cache miss. If we do not find and address then I call it as a cache miss. If I do not find a page I call it as page fault right, why can not call it kage page miss? And cache fault why should I say cache miss and page fault think about this why I say it is cache miss? And that as page fault why can not I say page miss and then cache fault? So, what is the different between a fault and a miss ok.

We will go to the is it all fine did you all follow? So, this is there are n number of such cache coherence photo calls like you have a machine you have a CPU for your mobile phone you have a CPU for a desktop like this. And you have a CPU on a server class what is the different? They have the same instruction sets, they and the same executable, but what would be the major different? The different would be one of the major different would be the cache organisation right. And so, what is the change in the see the way cache is organised this completely victual, to the complier completely victual to the user correct. So, because cache even if there is no cache your program will work, but it will work show if I completely go an invalid at the cache and I remove the cache I disable the cache suppose I have facility to do that still your program will work every time it will come to memory and do that. But So, so everything depends up on the cache, in terms of getting very good performance, especially in high n computing or high performance computing or whatever call. So right, cache architecture make or destroyed a service.

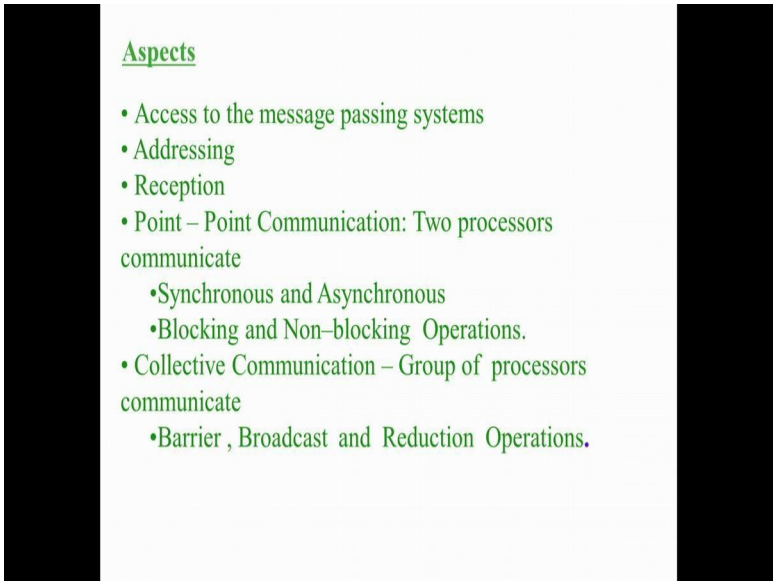
Now, we will go to the last part of the this course which very interesting part. Now we will now talk about parallel programming ok.

(Refer Slide Time: 22:21)



So, let me just ah we will just talk about bit of computing here, but then we will go and look at lot more on the program in ok.

(Refer Slide Time: 22:41)



So, what happen is let us let us I will I will cover whatever I am going to say now I am broad detail in my short of last 2 lectures right. Where I am I am going to motivate you to

what hells is pending in computer architecture. I want to cover a bit of history of computer architecture before I have end of the course. So, I would have started with that that would be boring. So, I am pushing it to the last and trying to keep it much more

interesting. Because you start appreciating lot of things of why certain things happen after taking this course. You will start appreciating history after you understand current day computing and the challenges. So now, what happen in say 2002 to 2005 the power consumed by a system is directly proportional to the frequency of the operation. It is also proportional to the voltage of operation.

Now, as I keep increasing the frequency my power consumption started increasing and started heating a wall, the wall essentially the wall in this case is that chip started actually burning right. So, I started (Refer Time: 24:08) constructive computing I add a distractive computing the, and this was demonstrated in several places where you switch on this system laptop actually burned. There are photos of laptop actually getting burn because of power issue. The interesting thing that happened here was that, if I had a you a program that will run on a single core, and I take A 2004 machine that will be faster than the 2010 machine, correct? Because if I had a program that will run only on a single core. In the 2004 machine it was running on a 3.5 giga hertz processor. Now in 2010 machine it on running on A 1 point gigahertz process correct. So, what happen? The program use to run very fast in old day machine now it is as now I am investing more money, but my programs speed is getting low right, that essentially forced people to go and start looking at multi core programming.

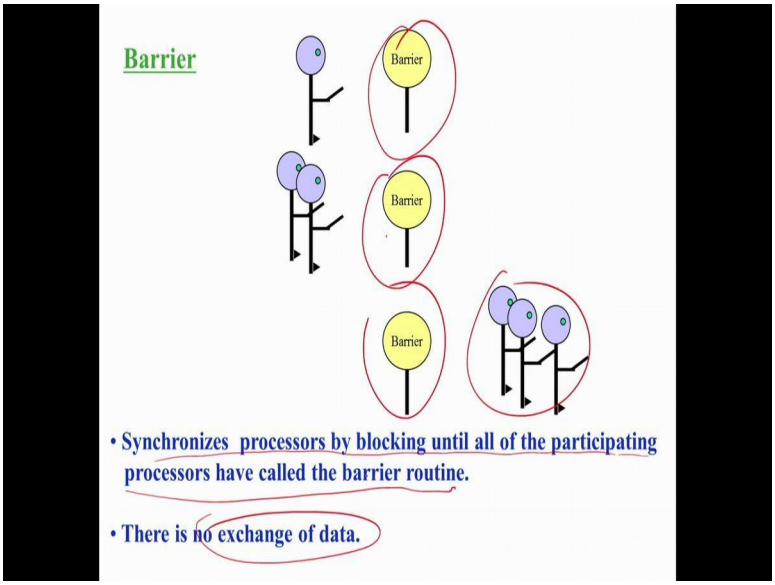
So, what you call as parallel programming right, when I want to do parallel programming essentially; that means, that I have a task I want to split it into small task one fellow doing all the job I will make 2 fellows do part of the job. And I will take back the result correct. So, if I had hundred numbers instead of one fellow performing 99 additions if I have 4 fellows each can perform 24 additions and then fellow can perform the 4 additions and give the answer and what do you get here I get the speed. Now I improve the speed 4 times right, instead of A 1 processor handling 99 additions there are 4 processors concurrently handling each 24 additions and then one more processor handling the remaining 3 additions.

So, I get immediately that speed, but when I want to do this right, one of the important thing is that if I have a task and I am splitting it into small task and giving to each or each one of the processor and asking all of them to do concurrently and giving me the answer. I now see the parallelism. I now see the speed up. But then these processor have to talk to each other right, they have to sing there operation. I split the task into 4 parts is not

that all the 4 can work in isolation. I have to stop at intermediate places and exchange information. And the way to we exchange this information essentially, forms the building block of this whole thing right. If I exchange the information through memory then it becomes a shared memory architecture.

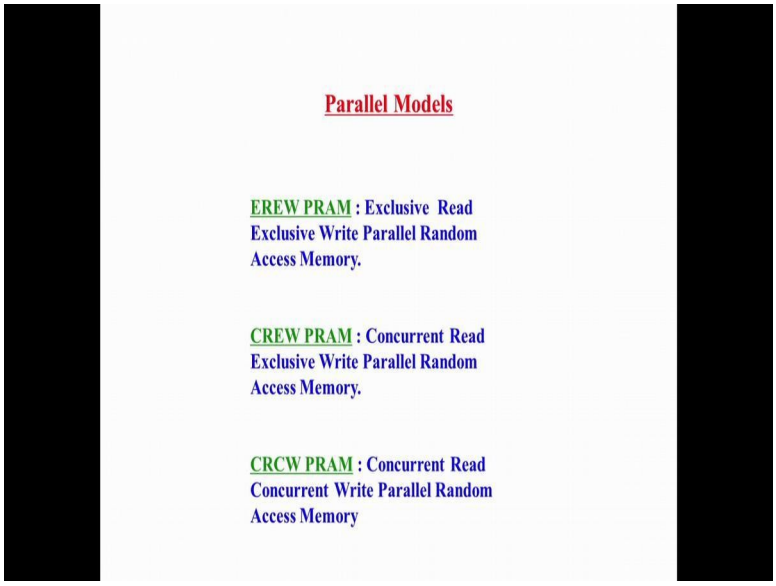
But when I want a exchange information in need to basically have certain constructs by which I can do this in a effective manner.

(Refer Slide Time: 27:36)



Let me motivate this Now, we will look at things like I want to just talk about 2 things, I have you to just talk about one aspect here which I call it has barrier ok.

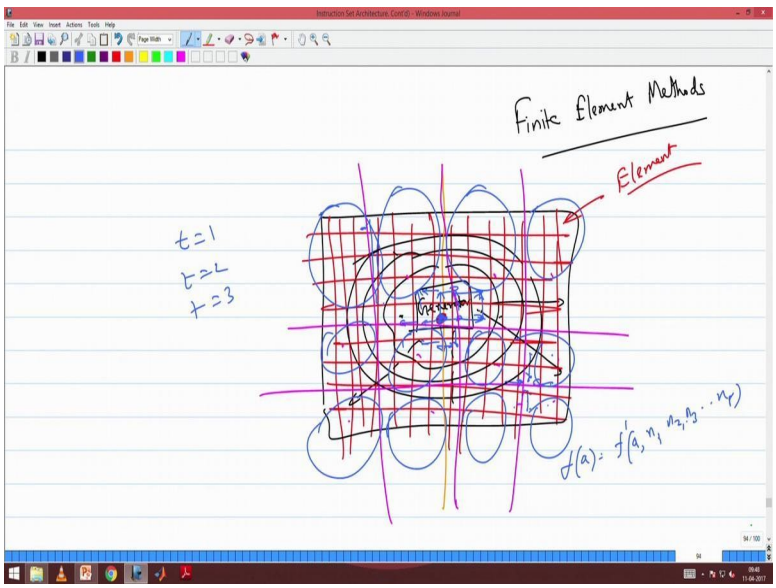
(Refer Slide Time: 27:49)



Let me just you know talk about barrier and then we will go to this parallel models ok. Today I talk about barrier and then in from tomorrow's class from next to 2 classes I will basically cover the parallel models of computation.

Now, let me explain barrier, one of the most famous you know parallel computing beneficially or is the flied of what we call as finite element methods.

(Refer Slide Time: 28:14)



You talk to aerospace you talk to bio technology, you talk to a b c civil engineering, will talk to d (Refer Time: 28:29) e electrical engineering ok. You talk to all this people one

thing that you learn is that all do some short of finite element method. Especially mechanical civil aerospace, navel, ocean, material science right. So, what is this finite element method. That see let us take let us take some simple problem like ok.

Now, this is one floor roof top. On top of it I am keeping a generator. This is a power generator. Now I am switching it on. Now this will start shaking. I want to see how that shake propagates through the all this direction right. When I stand here I have vibrations more vibration as I go here the vibration. So, I can form circles and as I go from the inner more circle to the outer more circle my vibration will decrease right. Now I want to study what will be the vibration here vibration here, here also. Why this I could see whether the floor will fall down? If I put that I want to stimulate right. So, what I do is this entire floor I split it into small small (Refer Time: 29:55). Each square here is called an element.

As I keep shrinking this grit size more and more elements will come, but I am now essentially partitioning there is a different between division and partitioning, discrete mathematic did you learn what is the different between division and partition of a set. Set s is divided into d 1 subset d 1 d 2 essentially means d 1 inter section d 2 did not be necessarily 5, but if d 1 inter section equal to 5, but if d 1 inter section d 2 equal to 5 then it is a partition. So, there is a difference between the word partition and division. Now I am actually partitioning the entire thing into small small elements, finite in number that is why it is called finite element.

Now, what I do? Now I have mathematical equation which can simulate vibration here I am doing it in blue colour, it will simulate a vibration here, and which will tell you how much it is spreading to this. After that from here this will spread to it is nearest neighbours. Slowly I will keep seen how this vibration splits. So, I analysed element after over the every time frame t equal to 1, t equal to 2, t equal to 3 whatever the time unite every time unit I will see what is this state of every element. A state of an element depends up on it is influenced by the state of it is nearest neighbours and itself ok.

So, let there be a function f of neighbour a is actually dependent up on some f dash of it is own self a and it is neighbours n 1 n 2 you have 1 2 3 4 5 6 7 8 right, there are sorry one 1 2 3 4 5 6 7 8 n 1 n 2 n 3 till n 8. So, a time t equal to 1 I evaluate this then time t equal to 2 I can evaluate. So, every time what I do I take my self. I take the values of my

nearest neighbour complete take a function and I put that value and then again I will do the next step right. More the number of more finer is this grid what will happen my accuracy increases. And you already see that this is embarrassingly parallelisable right. So, what I can do is I can split it in to what colour here colour ok.

I can split it into 1 2 3 4 5 6 7 8 9 10 11 12 processors. So, each processor one processor will response for this P 2 will be responsible P 3 P 4 P 5 P 6 P 7 P 8 P 9 P 10 P 11 P 12. So, everybody will start computing right, but please note that at the end of one time step, everybody has to wait for everybody else. Because this end boarder point right, when I want to compute for t equal to 2 it needs that t equal to 1 value of the edge. So, let us say we 8 are computing this when I finish my part I have to wait for you to finish the other part So that I can get the latest value. So, every processor is working independently, but when I finish the part I have to basically wait for all of you to finish and then I can go to the expand. This is basically called as barrier.

So, I put a barrier I put a barrier. At the end of the every iteration. What and what will happen? When everybody hit is that barrier then everybody will start. Now tell me how can implement barrier? Where are we seen something close to this in this course fine. So, atomic instruction what are atomic instruction. Instruction that cannot be stop in the middle right. So, this is basically atomic instruction. So, we have seen test and swap couple of instruction I have basically introduce. And we show that how the producer and consumers can sink, where using this atomic instruction. I give you a small algorithm I tough in this class right.

So, so barrier can be basically implemented using atomic instruction. So, the so, that is one very important thing. So, before we windup today. So, this is how this is how barrier works there are 3 processors I put one barrier here. So, when this is this is a barrier I put in all the 3 programs, when one somebody come a barrier he will stop this fellow is coming to the barrier stop this fellow also come, when all the 3 fellow reach the barrier all the 3 will moved. And the way implement the so, the barrier basically synchronizes processors by blocking until all of the participating processors have called the barrier routine. And there is no actually no exchange of data.

So, barrier is one particular construct that we need to build when many processes are working on that right. With as an basic building block I will now go an start teaching you

the PRAM, which is the parallel random access memory. So, this is very important I will give you some clues of how to go about programming multi processor architecture.