

**Parallel Computing**  
**Prof. Subodh Kumar**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology – Delhi**

**Module No # 02**

**Lecture No # 08**


**Open MP & PRAM Model of Computation**

(Refer Slide Time: 00:29)

```
#pragma omp parallel shared(a, b, nthreads, locka, lockb) private(tid)
#pragma omp sections nowait
{
  #pragma omp section
  {
    omp_set_lock(&locka);
    for (i=0; i<N; i++) a[i] = i * DELTA;
    omp_set_lock(&lockb);
    for (i=0; i<N; i++) b[i] += a[i];
    omp_unset_lock(&lockb);
    omp_unset_lock(&locka);
  }
  #pragma omp section
  {
    omp_set_lock(&lockb);
    for (i=0; i<N; i++) b[i] = i * PI;
    omp_set_lock(&locka);
    for (i=0; i<N; i++) a[i] += b[i];
    omp_unset_lock(&locka);
    omp_unset_lock(&lockb);
  }
}
end of sections */
```

Find the Error

Assume:  
variables declared  
locks initialized



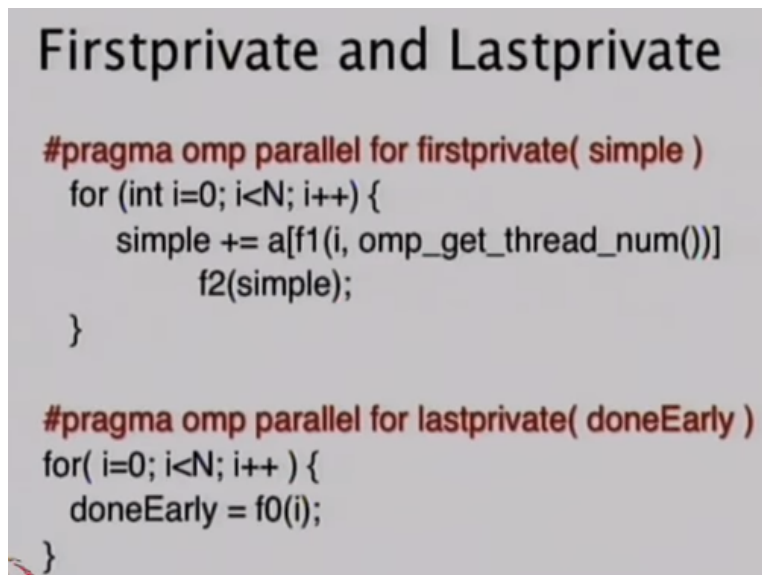
Alright here is some piece of code open MP code since I have read the back thing I hope you can read the front thing let everybody find out the error slash errors. Anybody still needs more time compiler is why should compiler read an error not but the compiler does not necessarily know what OMP said locked us does not know he semantics. Okay so as many of you as figured out hopefully all of you figured out there is be a deadlock there can be a dead lock.

Because one locks A and then B without releasing it and the locks be and then looks for it tires to lock it actually this guys locks it a while this guy has locked B then both of them are waiting for each of them locks right. Each other to release the locked they acquired but they cannot release because they need another lock before they are going to release anything okay what so that the more clear errors is also shuttle errors which again depend eventually whether or not error surfaces because after all in this case if you dead lock what is going to happen.

Because of non-flushes that is yeah if the dead lock does not and that happens that is your semantics session may be that is what you want that either this should get added to that or that should get added to that which got added first you do not know. It is not necessarily an L it might be because I do not know whether I have no idea what these variables means so I cannot say but there is something I can say looking at this code.

And that is the variable I the variable I is declared outside and it says the variable declared if the variable I is declared outside then both are them are going to change the same now you may be able to get by if it is in a critical section if the entire far in a critical section then one guy was from 0 to N and then next guy set to 0 and goes to N again okay.

(Refer Slide Time: 04:37)



```
Firstprivate and Lastprivate

#pragma omp parallel for firstprivate( simple )
for (int i=0; i<N; i++) {
    simple += a[f1(i, omp_get_thread_num())]
    f2(simple);
}

#pragma omp parallel for lastprivate( doneEarly )
for( i=0; i<N; i++ ) {
    doneEarly = f0(i);
}
```

Here are some example this is the example that the first one we have already seen the in fact the second one also we have seen but I had a non-confirming loop here I had a I less than N or done early. So I basically had removed that extra condition and now this is a proper loop but it is still doing the same thing it is saying for I = this is inside a parallel for. For I = 0 to 10 I want to evaluate something on I some function as 0 okay.

And inside that function of 0 for some reason you may quit early you may not necessarily evaluate it all the way to the end. Okay at the end of it because it is last private what will that done early contain whether or not the Nth or N - 1 or the third with which does did I = N - 1 got

done early okay. Now there may be a there is some hidden semantics here which says that who done early means three knows that we do not need that three it was done.

So 4, 5, 6, 7, 8 were not required to be done okay but if three was done 3 was going to determine that it was done early 4 would have done the same 5 would have done the same 6 would have done the same and - 1 would also figure out done early true right. So 0 to Y, 0 to sum one of the I values would have computed it I equal to that value + 1 up to N -1 would not have and if I only want to know if it got done early if I am interested in where I got done.

Then this is going to work just fine because if the last would have done early then either the last one was the one that got done or one of its predecessors got done early either way it go done early. Right again this is the little contrived so that is may be reason you are finding it hard to digest everybody is trying to do something okay and during doing that something they realize that they actually did not have to do it.

For example everybody is trying to look for some bad guy and they determine looking around the context around it that the bad guy is has already been found okay. So  $I = 0$  looks for it does not find it  $I = 1$  looks for it does not find it there have been parallel. So 0 did not find it at the same time 1 did not find it at the same time 2 did not find out at the same time 3 found it. Because 3 found it that internal logic says everybody else would not set anything but if 3 found it everybody else would also have found it.

That is the semantics it is that is the result done early yes listen to my story then ask your questions everybody is evaluating done early and if I evaluate done early = true that means everybody  $I$  greater than equal to my value who is guaranteed to evaluate it to true that is my story you write your own okay. There is inside of 0 okay so that is the semantics whatever they are doing if I figure out that done early was true then I know that I after me is also gong to figure out done early = T okay.

And at the end of it all I want to know is was done early true for anybody if it is true for the last guy it is somebody true for somebody so that is all it is saying. I just need to check is the last guy

saw the done early = true because it last guy did it then somebody before it may or may not have found it but at least we know that somebody did. Last iteration okay that is my story I am sticking to it.

(Refer Slide Time: 9:53)

```
Ordered Construct

int i;
#pragma omp for ordered
for (i=0; i<n; i++) {
    if(isGroupA(i) {
        #pragma omp ordered
        doit(i);
    }
    else {
        #pragma omp ordered
        doit(partner(i));
    }
}
```

Here is an example of ordered construct pragma OMP sorry ordered always comes in size and it is inside section or for. So it is says pragma OMP for ordered for  $I = 0$  to  $N$  if I am member of group A doing it pragma OMP ordered otherwise do it to the partner whatever the partner meets some other. Partner I evaluates to one of the other indexes so yeah hope I am assuming that is group A is uniquely going to identify everybody.

So people are member of A and B and C and D either you are member of A or you are not so those who evaluate to group A will do first ordered those who do not will do the second order anything wrong with it tow are actually linked they are all going to be sequential zed what that says is if I have reached an ordered then every index before me has gone out of the ordered of not the ordered of their ordered okay.

In fact even not there ordered they have not going to come here in this ordered okay so that is the more accurate way looking at it they are not going to encounter this order okay. It is there is still a guarantee that is the things these are not different sections is basically saying that section is that

part of code is to be sequentially this part of the code is also to be done sequentially that means these are together to be sequentially does not say that this is semantics.

So either of them one yes that is precisely true this is not allowed because both of the ordered have to be it is part of the same ordered because there is section ordered section right anything that comes in the ordered group when you reach there you have to make sure that all in anybody before you is not going to reach there okay. Now if I have first ordered second order is part of the same section it is not that I have to cross ordered and there is a new ordered.

So this ordered and that ordered are I cannot be at both place at the same time which means that I cannot get there and here which both required 0 to appear first there 0 lets us talk about index wise index 0 has to go through this first and as to go through that first and hence this is not going to be that you have as a programmer have to make sure. This is definitely an error it is yes in this case it is syntactically determined right.

If there is an IF statement it may or may not warn you the short story is both they do and do both of these things are ordered part of the same ordered okay. Everything thread is allowed to encounter at most one order if you do not encounter an order you just go on okay. If you do encounter an order then it is all the order as a part of the same although you know not executing all pieces of code in this case if you are in group A you are not going to do the second half of the code that is also the part of the same ordered.

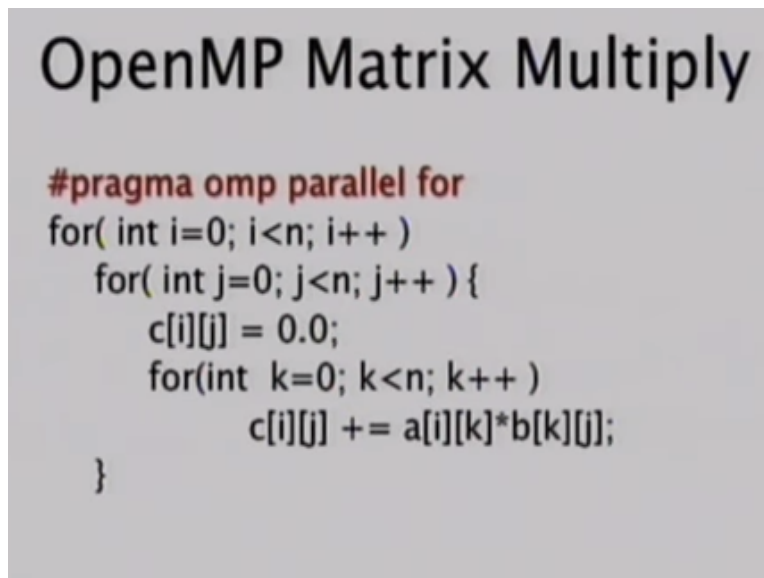
It is like on ordered section you may execute some piece of it but that ordered section you can get to only if all index is before you have got it okay. No run time is basically syntactic two ordered next to each other without an intervening parallel for OMP parallel are not just allowed. But this is not well not intervening parallel like this would be allowed because they are not next to each other in the same block.

They are in different control paths that what compiler does it figures of the control flow graph and says this is one block of code this is another block of code and there is jump between them okay. So if these happen to be in the same block of code that is not allowed so given what I told

you if there is an ordered construct let us look at this again. This is inside this OMP for would make sense only in OMP parallel yes I am going to modify this code a little bit am going to remove the IMP pragma ordered in the else part what do you thin below?

It is allowed actually because it does not allow who is going to evaluate to is group A may be everybody values to remove group A nobody goes there exactly. So it is possible that some buddy does not encounter the ordered clause at all he will proceed but any index after it which is waiting for the ordered clause and asking if that guy part here or not is never going to get pass that one okay.

(Refer Slide Time: 18:43)

A slide titled "OpenMP Matrix Multiply" showing a C++ code snippet. The code uses the OpenMP parallel for pragma to parallelize a matrix multiplication. The code is as follows:

```
#pragma omp parallel for
for( int i=0; i<n; i++ )
  for( int j=0; j<n; j++ ) {
    c[i][j] = 0.0;
    for(int k=0; k<n; k++ )
      c[i][j] += a[i][k]*b[k][j];
  }
```

Alright matrix multiplication simple example get rid of the parallel OMP pragma for and that is regular N cross N matrices A and B are getting multiplied and the results is gain I want to parallelize is I just put that thing at the top what happens how does it work is it correct or not. So the top for is really the for struct right and so for some different I is the variable for and different values of I will allocated to the different threads in this case the default is static number of ahh chunk size is not been given.

So everybody every thread is going to get one fresh some set of I one over the number of processors are I have or N over the number of processors of I and then they will compute so many rows of the result see what if I have lots of processors right now put another for. So when

the first get encountered you create some threads second gets encountered each of those rows will get broken into blocks.

Now everybody is computing one block of data one position if we have got  $N$  cross  $N$  processors right. If it is  $N$  cross  $N$  it is fewer than  $N$  cross  $N$  then one block I have not said what num threads should be. The number of threads multiple of existing threads not sure I understand you question. How many threads should be? It depends on the task at hand it if each thread as lot to do then you do want more threads.

Because when it had lot to do there is more likely chances of it getting cache misses it will get waiting for something to happen if it is just addition of two things then probably they are all waiting the same amount and so it does not necessarily makes sense to do lots or for that matter if there is just compute intensive you have iterating over the same piece of data small set of data again and again and again.

In that case you are probably just running the CPU as fast as I can in which case nobody is waiting in which case you would want as few threads as number of processors but again in general for most applications you want to be somewhere between 5 to 6 may be even 5 to 10 I would say times for Intel type architecture if you are talking about GPU type architecture then we had going 10 to 100 may be it does not context which GPU okay.

So you can do this also okay and have every single element being further for every single element or block of element being further broken into all the products that has to do with the products being reduced by a sum okay. I am going to stop here now talk about model of computation today so we do models of computations we do it even in the sequential the idea is that you can simplify you can write that means something you have the notion of an underlying obstruct level architecture.

Without worrying about what the actual instruction set is how much memory I have and think so that's all carrying over the same thing to parallel computation work.

**(Refer Slide Time: 23:48)**

## Models of Parallel Computation

- Simplify specifying, reasoning, analyzing algorithms
- Must abstract away many details
  - Should predict computability
- Should track performance
- General classes
  - Shared Memory vs Distributed Memory
  - Synchronous vs Asynchronous

You want to be able to specify programs in this model reasons that they are correct argue that they are correct figure out how many steps they take okay. Once we determine what is step is. So all of those are going to be the realm of model of computation that is why we typically have the model of parallel computation that people can develop algorithms in this model and then you can port it apply it to a given architecture.

Of course over time the variety of single processor architecture has restricted itself to a small set and that has not quite yet happened and to the parallel architecture although it is beginning to happen an extent but irrespective of all that you will find that if you specify an algorithm in the PRAM model it make sense to implement in some other architecture which is not quite PRAM and I will tell you what PRAM is shortly.

But the idea is to abstract away the details so that you are worrying about the just the main high level structure of the algorithm one thing you would want because you are want to analyze performance is that if you say that it is order  $N$  takes  $N$  squared steps it means something in the real world also okay. To a large extent even that is going to be true in spite of the fact that we are going to make very gross assumptions we will in general only these two varieties of things right.

You have got shared memory or you have got distributed memory and in typical modern parallel computer you have a bit of both so we will want to have model that can model shared memory

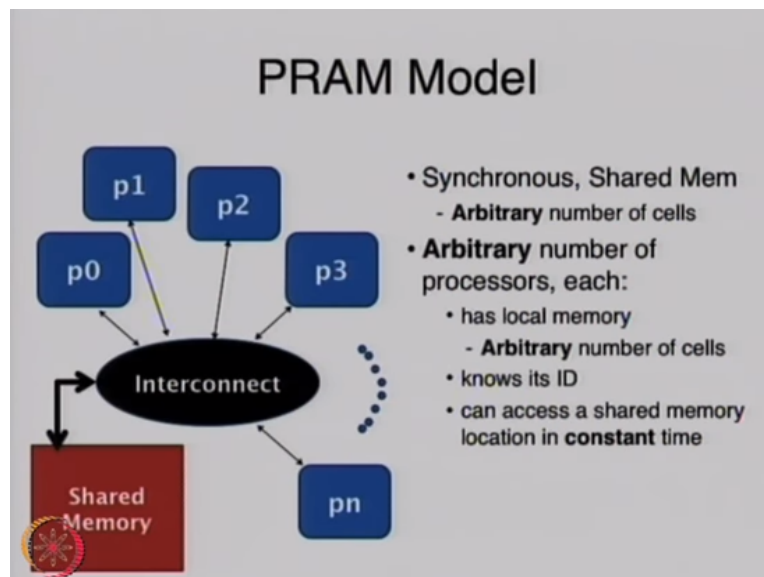


architecture as well as distributed memory architecture we will look at how the model that we are going to study first PRAM model does that but eventually people come up with different models one representing the shared memory one representing the distributed memory architecture.

So if you look at that space also little bit the other dichotomy that you notice is synchronize versus asynchronous. For example the PRAM model is a synchronous shared memory model that says is that all the process somehow have a shared clock and they are proceeding in a lockstep fashion not necessarily SIMD but they know what happen they know that I am at step number 39 asynchronous is that at their own speed my step number 39 has nothing to do with my step number 39 which would be clearly more realistic.

But turns out to be not quite necessary although there are asynchronous memory shared models as well as synchronous similarly they are asynchronous distributed memory models as well as asynchronous we will probably not talk to much of asynchronous because it only complicates analysis we talk about it as a model but probably never going to use it.

**(Refer Slide Time: 27:30)**



So here is basic PRAM model which is extension that the RAM model that you have seen in the sequential in a here there is a memory that is shared by a number of processors through some interconnect is essentially painted black here it is a black box we do not know how that

communication to shared memory happens. We just have a bunch of processors and a bunch of addresses which can be globally accessible to all of them okay.

In addition there may be some local memory each processor may be the cache or whatever something that is not accessible to the other people okay. It is not shown here it is a part of P0, P1 and P2 and so on. Just like in RAM model we do not say there is so much memory right this as much memory as you need is the same story yet as well as many processors as you need there is as much memory assumed.

There is as much local memory as you need there is as much shared memory as you need so that is the first abstraction I would not say assumption but it is an abstraction will talk about how it affect our analysis. Yes we one is the amount of resource how much memory I have how much local versus or shared how many processors I have here there is their speed each shared memory access can be done in a constant amount of time which is exactly what assumed in the PRAM model okay.

Now it is local or it is in cache or the like okay any processor that wants to access any piece of memory local or shared it can do so in a single unit of time whatever that unit may be order on. Also each processors known it is ID so locally it determine the its processor number 3 of whatever the size is. So what are the unrealistic parts here? Shared is one memory location one addressable memory location constant access is almost never going to be possible.

Especially if you have anything more than 6, 7 processor even 6, 7 processor is not really constant may appear constant. Fixed in fact it will be even more clear when we take a little bit detail constant is clearly bounded right. So when you say it is bounded with no reference to the size of the problem it is constant so it can be 1000 hours it is still constant. Well when we say that the number processes is unlimited we cannot guarantee in thousand hours every single processor will be able to access every piece of sure.

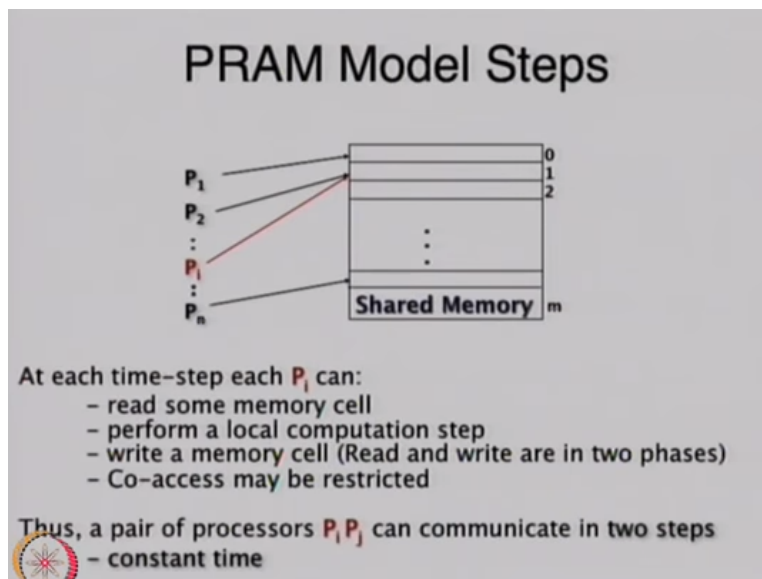
Because the number of processors may be many more or many powers of parallel if you fix it is possible to make it constant right but we are not fixing it. Using we assume that as many

processors as we need and they are all able to access the same shared memory in a constant that is not what we need it means that when there are many processors they can each tell but I have wait for 1000 hours to get to this exactly not a bound.

Well the bound needs that they can then pretend that they got it in 1000 hours even if they got in an 100 they will sleep for the remaining 900 hours. So it is unrealistic both because of constant assumption and the amount of resources are assumption constant is little bit order to simulate on the other hand the truth is that you will have a bounded number of processors in reality and that will make the memory limit the memory size or the access latency constant.

Also it would not give you real performance because when just like I said if you say that 1000 hours and you only can get it in 100 you are sleeping for 900 hours that is not really what we do. If you can get it in 100 you will want to do something useful for that those 900 hours if you do then the predicted performance would not necessarily be the same as what you are there okay. So that constant will actually be a bound and it will be a constant because sometimes you will fetch it from somewhere nearby whereas sometimes you will fetch it from somewhere far away okay.

**(Refer Slide Time: 33:19)**



So here is a little bit more of the detail got am not showing the local memory here local memory was just like it does not a sequential processor okay. The shared memory is accessible by any number of processors here am assuming that there are M cells whatever that M may be depends

on the algorithm not on the problem not on the given architecture it depends on the algorithm actually.

Which in turn may be depend on the problem the number of processors and each processors can access some memory location and a time step is defined thus in every time step you can read some memory cell do some computation okay constant amount of local computation and then you can write a memory cell. Those three steps together constitute a step either of them you could think of it that.

You can say these are three steps we can do that here it is combined so that in that time step you are completely encapsulating that piece of work again to read something do something on it and write it back okay. And it is not entire three step is atomic but each of those is atomic you do not need lock here not only cannot have lock you do not need locks. That is one of the simplification is that you determine how many you are going to assume right.

So you say I are you want to give me  $N$  things to sort I will tell if I had  $N$  processors I would able to sort in  $\log$  of  $\log$  in time that is not right but just for instance I can say if you give  $N$  square processors I have sort it in one line constant okay. It depends on the algorithm but the algorithm  $N$  will depend on the problem in turn. Not here right we are abstracting here the architecture so the reasoning and analysis is going to be in the PRAM model and then will see.

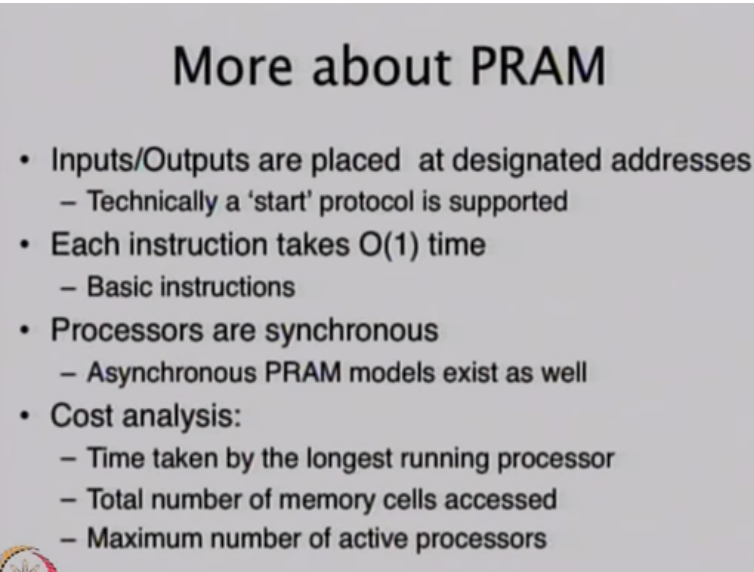
It does eventually have to be implemented on architecture it has to map to a architecture and somehow it needs to make sense for all architecture at least variety of architecture if not all okay. So you read together then you are reading all the processors are also reading and you are writing all the other processors are also writing and so it constant amount of time you can communicate data to somebody constant sized data.

You write something and the next step you read something I will tell you what co access is restricted means very shortly. Co access is restricted only means that there are varieties of categories of PRAM models where you may not allowed to read ahh in common meaning that

two cells above the two processors cannot read the same cell in the same clock okay or cannot write in the same cell in the same clock.

So noting is deferred it is just that them in the model your programs are illegal if you write into the same thing okay simply cannot write such that is detail of the architecture. But yes at the abstraction level they all writing to some locations and in some models there is locations may overlap may have some commonalities in some model they must be independent unless in the same clock at the same type step they will all be okay.

**(Refer Slide Time: 38:26)**



## More about PRAM

- Inputs/Outputs are placed at designated addresses
  - Technically a 'start' protocol is supported
- Each instruction takes  $O(1)$  time
  - Basic instructions
- Processors are synchronous
  - Asynchronous PRAM models exist as well
- Cost analysis:
  - Time taken by the longest running processor
  - Total number of memory cells accessed
  - Maximum number of active processors

Alright the input and output in these are some of the technical requirements how does it read inputs how does it produce the final answer how do processor gets started right it is right I have many number of processors I need how does the algorithm say I need processors or  $N^2$  processors those are technicality those are assume that  $N^2$  processor are active if we need them.

But there are ways but there are ways in which you can always have processors 0 startup and its activate the other processor and then you have to add the count that the cost of activation in the total cost of algorithm right. If you are using more processors it will take more time to activate them and so on. There is also notion similar to RAM again then there is some default instruction

set which we are not going into it like add subtract basic operations that you may assume that can be done in order one or assumed here right.

So although am not going into that level of detail of what is instruction set assumed it is basically the same as RAM. You cannot say that one of my instruction is sort and things okay because you can do this in order in this model so that is not allowed the instructions are basic instructions whatever we have no understand as basic instructions. I already said processors are going to be synchronous meaning that all going to be reading then all going to be doing computation then all going to be writing okay.

And asynchronous versions exist which then requires some synchronization at some point in terms of the cost clearly how many steps are needed will determine how expensive your algorithm is and does how many such steps of read modified write is needed similarly cost of how many processors you are using is going to be important and somehow those two may be needs to be combined into one number that says that is the quality of the algorithm.

And of course the space that is typically related in a trade-off sense to the amount of to the number of steps need to take. But again just like especially in the beginning we do not worry too much space you are using we focus mainly on how much computation you do. You are going to do the same thing you are not going to worry about how much shared memory we are going to assume.

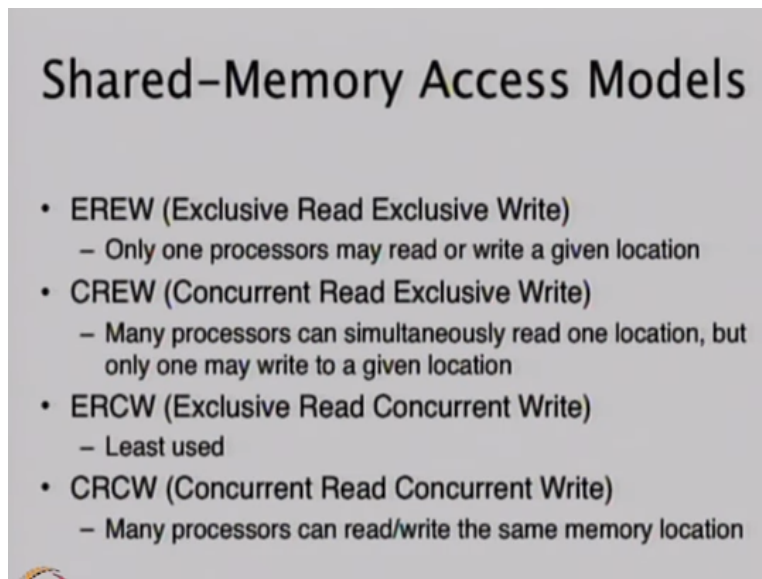
You do not worry about how many processors we have and how much computation they do okay. Here are the restrictions that I was mentioned E stands for exclusive and C stands for concurrent okay. So one variant of the PRAM model is EREW PRAM model which means in that if that is what you are assuming then read as to be exclusive we need two people cannot read from the same location at the same time and a write as to be exclusive similarly two or more.

Processors cannot write to the same location in the same clock okay so you can read but you cannot write CREW model. Concurrent read but exclusive write okay that is synchronize model right everybody is an read step a compute step or right step. So that possibility does not exist in

PRAM that some is reading and some is writing okay. ERCW is just for completeness here you never argue anything in ERCW model because it is completely unrealistic.

You want exclusive read but everybody can write together that is allowed you are not going to talk about that after this slide. One location once right because otherwise basically one processor is reading in one clock that would make sense.

**(Refer Slide Time: 43:06)**



## Shared-Memory Access Models

- EREW (Exclusive Read Exclusive Write)
  - Only one processors may read or write a given location
- CREW (Concurrent Read Exclusive Write)
  - Many processors can simultaneously read one location, but only one may write to a given location
- ERCW (Exclusive Read Concurrent Write)
  - Least used
- CRCW (Concurrent Read Concurrent Write)
  - Many processors can read/write the same memory location

And this CRCW is concurrent read concurrent write okay this is the most powerful model of course requires most amount of hardware also meaning that two processors can write to the same locations also at the same clock.

**(Refer Slide Time: 43:30)**

## Concurrent Write (CW)

- Priority CW
  - Higher priority processor (normally lower index) wins
- Common CW
  - Succeeds only if all writes have the same value
- Arbitrary/Random CW
  - One of the values is randomly chosen

They are varieties of concurrent right ah and then you have got to assume one of them again it depends on what the underlying architecture provides probably. The variants are priority concurrent right meaning that there is an inherent priority and this is in fact one of the easiest to implement in some let me take that back not always it depends on the under the hardware around it.

But in some cases it is deceptively easy to implement it says that if you are a processor ID is smaller the you will whose processor ID is bigger does not get collate okay. So there right is just ignore lost the higher priority processor wins okay common is if they are trying to write the same thing then the right succeeds otherwise nothing we write nobody reads nobody writes okay. How do you know nobody writes how do you know whether the thing you wanted to write got written.

Next log we can read it that check and you will know this is relatively easy to implement arbitrary at last people are trying to write somebody right we will then again you can check who got to write. Yes correct not who got to write but whether wanted to write got written EREW is the least powerful CRCW priority is said to be more powerful. So this is increasing power what that means is that when you have priority you can simulate a CRCW model which is common.

If you had that can simulate a CRCW model which is random which can simulate CRCW which can simulate EREW okay. And it should not be hard to imagine how you would simulate right



how would CRCW, CREW lets say simulate and EREW what does an EREW do? It make sure that the structure of the program that only one is right okay. So EREW nothing happens only one person writes okay how do you simulate the let us say common with priority.

Common said that it will get written only if everybody is trying to write the same thing in priority you can write even if there is different and that winner you ever writes has always the lower priority. So if  $K$  processors of the  $P$  are trying to write to the same location the one with the highest priority will made. Without reading how can you write the program you do not quite get it that mechanism need not exist in common but we are not worrying about what that mechanism internally does one.

I have got a PRAM machine which is a CRCW priority PRAM machine okay somebody gave me a program that works in CRCW common can I run his program on this machine. Can I make some compiler modifications to his program? So will run on the machine in the program also it is allowed multiple people to write but only if all where writing in the same thing should it get written.

Here if they are write one of those they are the highest priority one will get it will something will get written even if they are all the same. I did not say how long it will take but yes it should be double if constant time simulation means basically what I will specifically I want a compiler to take that program and so something to it and run it on this machine so that the results ultimately or exactly the same as running back program on a PRAM that is the common CRCW.

It can add some more instruction in fact it has to add some more instructions each processors can read that so priority will everybody will try to write the priority one and then there will be in the next step it links a set of new instructions which will read it back and the given architecture is priority the program is of the common. If you know what was before that write you can write it back.

Before you write you have to read what is there so that you can undo that right so you have got the read done. This is simple enough it is good exercise I will let you think about it and then we

will talk about it again may be after the break or maybe the next class okay. CREW would be the simplest template oh wait I said that because we were running out of time I should not have gone there we are going to stop.