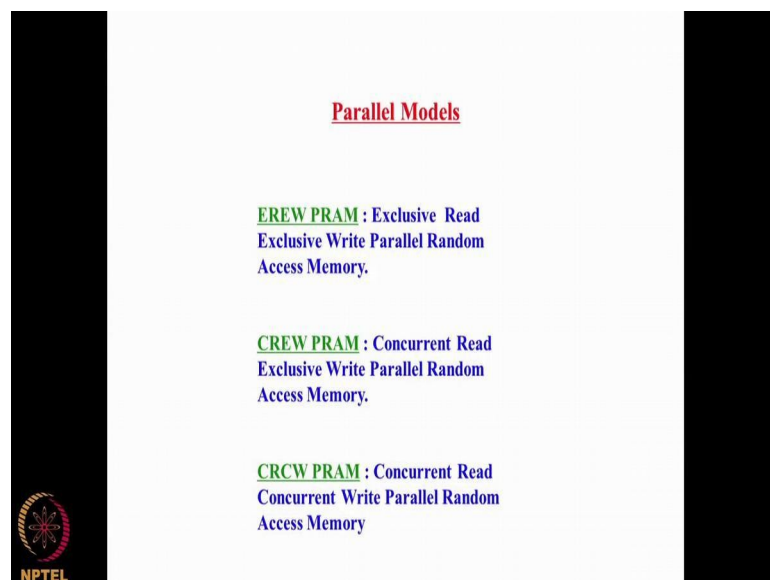


**Computer Organization and Architecture**  
**Prof. V. Kamakoti**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture – 40**  
**(Part -1)**  
**Concurrent Programming in Hardware**

So finish at the earliest, fine. So, we will do this part of the story. So, one of the most important thing that we need to learn is how to program a parallel machine.

(Refer Slide Time: 00:26)




**Parallel Models**

**EREW PRAM** : Exclusive Read  
Exclusive Write Parallel Random  
Access Memory.

**CREW PRAM** : Concurrent Read  
Exclusive Write Parallel Random  
Access Memory.

**CRCW PRAM** : Concurrent Read  
Concurrent Write Parallel Random  
Access Memory



Now, what I am trying to teach is some small tip bits of how to go about programing a parallel machine this is a completely algorithmic treatment, then there are courses on concurrent programing there are parallel programing courses high programing for high performing computing computers or high performance computing courses which deal with you know parallel programing that is part of your curriculum and electives. So, there are many many electives in the department the course on

concurrent programming will also teach you much more about this. But what I am trying to tell you is how can I model a parallel computer how can I model a computing on a parallel computer I think that is very much is necessary and I think it should be part of a computer organization architecture course.

So, like how there is an algebraic model of computation for your sequential computing single core computing right based on which you have got all your big one rotation size order a  $n^2$   $\omega n^2$  you did have those things based on that algebraic model of computation. Now there is a parallel random access memory access random access memory model of computation which you call it as the p ram model, p ram here stands for parallel random access memory. Now there are a set of processes processors there are all writing into the same memory correct.

(Refer Slide Time: 02:13)

Shared Memory Architectures

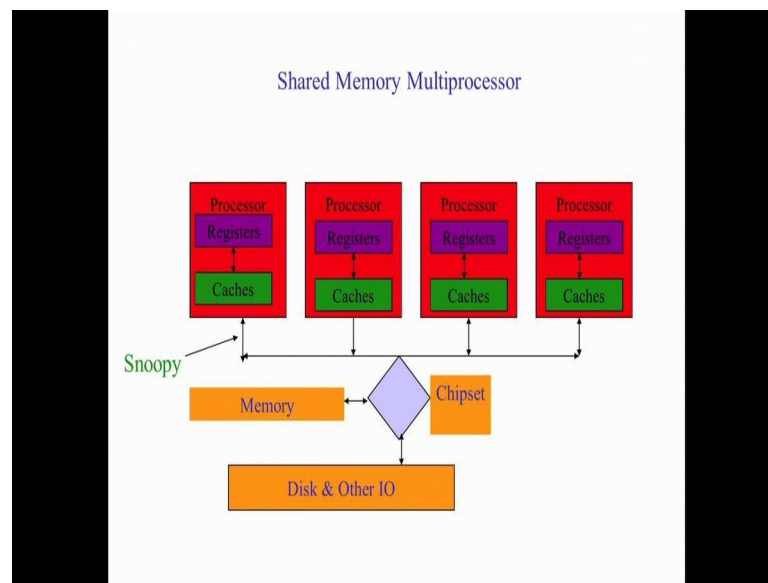
Example : step 5

	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				WrMs	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	WrMs	P2	A1			10
P2:Write 40 to A2							WrMs	P2	A2			10
				Excl	A2	40	WrBk	P2	A1	20	A1	20

Assumes Initial Cache State  
is invalid and A1 and A2 map  
to same cache block.  
but  $A1 \neq A2$

And there is a bus that is connecting this processor there as we are seen in the earlier part here right and every this is a symmetric multiprocessor system also meaning all processors are equal capabilities and they can read or write from any location in the memory right.

(Refer Slide Time: 02:23)



So, now, at any point of time there will be a token whoever has the token will come and access the memory I cannot say while I am writing the algorithm I cannot say who has the token. So, let us take one time stamp at that time stamp there are 4 of us, anybody can have a token, but at that time stamp I every fellow will have something to read or write from memory right at that time stamp every fellow will have something to read or write from memory. Who are gets the token first will write it write or do will call it as access the memory access is a word use for both reading and writing. So, everybody will have something to do with the memory who gets the token will access. So, when I run the program again let us assume that same time step or same step the token may be with somebody else also right. So, at some point of time, let us take all the 4 programs are running they are trying to do one job together I go and take some time slice at that point every fellow will have something to write into the memory or something to read from the memory.

If it is the case that at any time stamp I take, I will stop the algorithm no 2 fellows are reading or writing from the same memory location then it is called exclusive read exclusive write parallel random access memory modern right. I stop the program at any point of time I just stop it freeze you know there is a game called freeze. There are what happen now I will say who all want do what with memory then I find out no 2 fellow want

to read from the same location or no 2 fellows want to write into the same location then it is called exclusive read exclusive write parallel random access memory. So, if I

write a program if I write a program such that at any point of time I freeze that program I write a parallel program a multi thread that program where each thread will execute on one each of these processors I write a program in such a way that at every time step I stop a say freeze what will happen at any time I go and say freeze and I look at all the processors what they want to read or write no 2 processors want to read or write from the same location. So, no 2 fellow will read the same location also.

So that means, as processor p 1 I want to do some access it will be some location a one like that a 2 a 3 a 4 none of these locations will be equal they are all different then this is called a exclusive read exclusive write parallel random access memory model right. Now if that program is such that 2 fellows want to read from the location there can exist at more than one fellow who want read from the same location, but no 2 fellows will write into the same location then it is called a concurrent read exclusive write model.

So, there will be several this fellow wants to write to a 1 this fellow wants to write to a 2 this fellow wants to write to a 3 read from a 3 and this is read from a 4 a 3 and a 4 can be the same because they are trying to read. So, more than one fellow can read from the same location, but no 2 fellows write to the same location you can write to different locations then that type of a model is called concurrent read exclusive write model. The third one is any fellow can read or write from this any of the location then it is called a concurrent read concurrent write model. So, there are 3 types of models there are 3 types of models the EREW model which is exclusive read exclusive write where in I freeze at some point no 2 processes will read or write into the same location.

A CREW model where a freeze at a point 2 fellows can read one or more fellows can read from the same location, but no 2 fellows write to the same location. There is a third model where any number of people can read from any location or write into the location. So, the program that you write may satisfy one of these criteria. So, absolutely there is no issues in EREW absolutely there will be no issue in CREW also right because its only reading, but there can be one specific issue in CREW. What is that issue? Let us now start little more in (Refer Time: 07:43). Now we say deterministic algorithm what is deterministic why it is called deterministic algorithm, there are 2 varieties of algorithm right deterministic algorithm randomised algorithms or do you know randomised algorithm you have read randomised algorithm. So, what is a deterministic algorithm?



For are given a free condition the post condition is well defined for every instruction for every step in your algorithm then it is called a deterministic algorithm.

What is randomised algorithm? For every step the think is not well define; obviously,. So, toss a coin I do not know whether it will come and exact tails unless you read the coin right followed. Now the parallel algorithms that I am going to write should also be deterministic in the sense after every time step I should get the same result any number of time I am executing see suppose I am executing a program toss a coin right first time I execute it will give me heads again it will give me a tail. So, you do not know the output, but what we are interested here is to run deterministic algorithm on this parallel models right; that means, I run it once I stop it sometime t I see the results (Refer Time: 08:52) sometimes step t not time step I should get the same answer irrespective of whichever time I am running. So, one time I run I should not get one answer a another time I should not get another answer b that contradict the definition of determinism or deterministic nature of the program.

Now with EREW do you think there will be a problem with deterministic execution no correct it is obvious right with EREW will not have a problem with deterministic, but CREW will be have a deterministic no, with CRCW will there be a issue deterministic yes right the reason is - both of us want to write at the same point of time he is writing 2 I am writing one. So, one time when I am executing if the token is with him then what will be the final value one because token is with him he will 2 after the token will come to me I will write one right in another time the token is with me I will write one then you will get the token he will write 2.

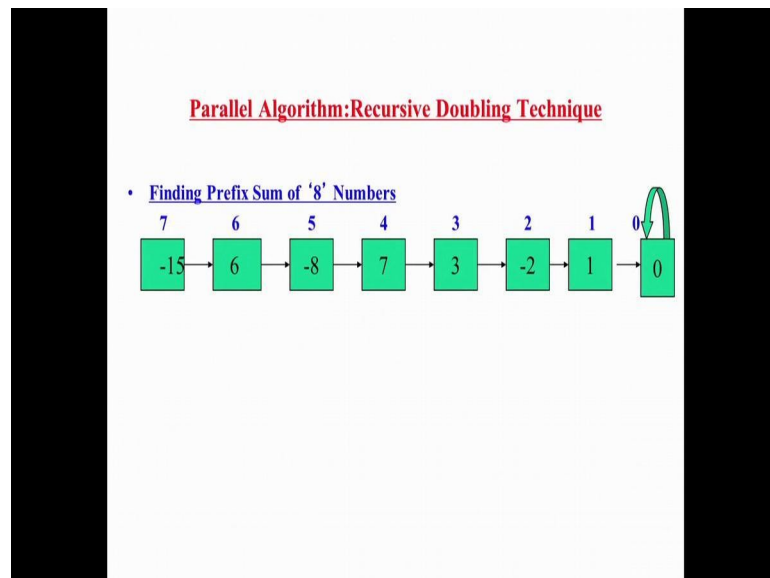
So, so at the end of this step one time it will be 2 another time it will be one. So, when I write an algorithm for the CRCW model correct I should see that at any point of time if 2 fellows want to write into the same location if 2 fellows want to write into the same location then they should write the same value right. So, when I am designing algorithm for the CRCW p ram model, I should ensure that if 2 fellows want to write into the same location then they should write the same value. (Refer Time: 10:47) are you getting what I am saying followed if you does not happening then what I told would happen right. So, he wants write 2 I want write one if he gets the token the answer will be one if I get the token answer will be 2 correct.





Now we will design algorithms for EREW, EREW just little bit larger extension of that. So, we will write one algorithm for EREW then one more algorithm for CRCW and that we will do today we will finish it today.

(Refer Slide Time: 11:19)



Now this is known to you the recursive doubling technique now I have 8 processors right numbers 0 to 7 of course, this 0 processor will just do nothing right I want to do the prefix some. So, I have 1 1 minus 2, 3 minus 2 plus 1 7 plus 3 minus 2 plus 1 minus 8 plus 7 plus 3 minus 2 plus 1 I want to go it. So, this you are already seen where did we see this recursive doubling technique where did we seen recursive double ah.

Student: (Refer Time: 11:48).

(Refer Time: 11:50). So, on what operator can we do recursive doubling?

Student: Associative.

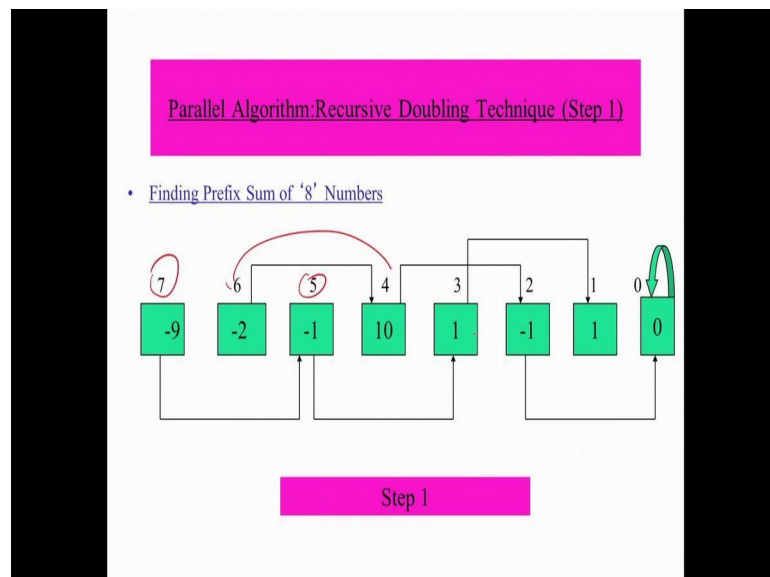
Associative it also called some high (Refer Time: 11:58). So, we saw this recursive doubling technique so what we do? We just add so every processor is given one location every processor is given one location and what it does it just it reads its value that is one time step, then its reads its value that it is pointing to that is the second step, adds it and stores it back where in the same location and if it is pointing to 0 then basically it does not read that

value it know it is a 0. So, 0 is put like a you know sentinel flow. So, what happens at the first step everybody will be reading its own value. So, it is exclusively

read right and in the next step it will be reading its neighbour's value in this case it is only reading nobody reads from the same location right this fellow 7 will be reading from 6, will be reading from 5, 5 will be reading from 4; 4 will be and 1 will not read anything 4 right. So, it's again exclusive read.

After that what it will do? It will add minus 15 plus 6 and store it here it will be minus 9 or whatever. So, it is again exclusive write. So, at every step I am doing only every processor reads from some location no 2 fellows read from the same location and they write to one location and no 2 fellows write to the same location. So, this is very much fitting our EREW algorithm.

(Refer Slide Time: 13:37)

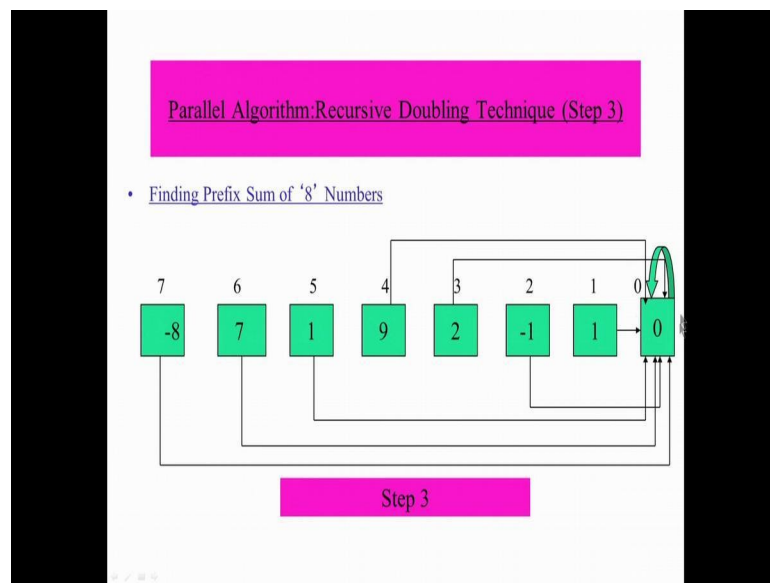


So, first step what will happen this is what happens everybody would have read themselves and then neighbour and this is how in the second step again you see this fellow will be read 7 will be reading 7 will be reading from 5, 6 will be reading from 4 and so on right. So, this processor will read minus 9 then it will be then it will read minus 1. So, when this processor is minus 9 this fellow will be reading of minus 1.

So, the processor will read from itself then where it is pointing to and no 2 fellow point to the same location. So, at the end of this, so second step also can be executed as CREW

similarly third step, similarly 4 step right and I do not read this 0 this just a centimetre.

(Refer Slide Time: 14:17)



So, you are this is sum, sum is a semi group operator what Raghav, sum is a semi group operator and, so this entire prefix sum can be carried out with in EREW p ram model are you able to understand what I mean right that is what I mean by steps. So, at the end of every step you do a barrier I have already thought you barrier right. So, read this barrier then do this barrier then write barrier. So, I can put barrier and then. So, that I synchronised such that no 2 fellows read or write in the same location and that is where you need a barrier to execute right are you able to get this, so this is one. So, what happens, how many processor I have  $n$  processor  $\log n$  time I can finish right. So, I had 8 I finish in 3 steps. So,  $n$  processor  $\log n$  time I can finish this. So, like this I can do prefix minimum, prefix maximum, prefix multiplication, everything I can do  $n$  processor  $\log n$  steps.

(Refer Slide Time: 15:13)

- Prefix Sum of  $n$  numbers in  $O(\log n)$  steps
- EREW Implementation
- Applicable for any semigroup operator like min, max, mul etc.

Processor  $n - p$ -processors  
Time  $O(\log n)$

Processor  $\times$  Time = Complexity of the best known seq. algorithm  
 $1 \times \log n \neq n$

So, now, I will introduce another small thing called I hope I have that I will introduce (Refer Time: 15:44) sorry I did not put the slide here. So, I will just see. So, what if I have if I have  $n$  processor I can do in order  $\log n$  time what I have  $n$  by  $p$  process, if I have  $p$  processors.

What if I have  $p$  processors? So, what I will do is what is the speed up I can get this is actually called as this is processor this is time processor into time should be equal to complexity of the best known sequential algorithm right. So, I have  $n$  processors  $\log n$  time is  $\log n$   $\log n$  is the complexity of doing prefix sum what is the complexity of the best known sequential algorithm for doing prefix sum what is the complexity of  $n$  right

So,  $n \log n$  is not equal to  $n$  right  $n \log n$  is not equal to  $n$ . So, this is not an optimal algorithm. So, what is an optimal parallel algorithm? An optimal or parallel algorithm is one in which the processor into time product is equal to the complexity of the best known sequential algorithm; that means what, you have equally split the work load among the different processors that what it that is what it means right. If this particular condition is satisfy; that means, I have equally split the work load across different processors right. So, in this case the pt product is  $n \log n$ , but the best known sequential algorithm will take  $n$  unit of time. So,  $n \log n$  is not equal to  $n$ . So, this is not an optimal

algorithm recursive doubling. So, how will I make it optimal? So, we use something which is very trivial let us see how it is happening. So, suppose I have  $p$  processors.



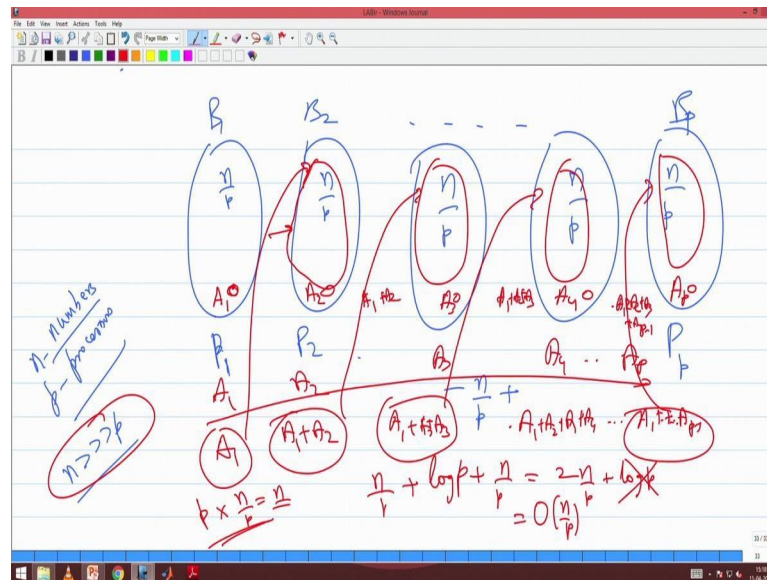
Student: Why are we quantifying complexity is processor (Refer Time: 18:04).

Because that is the total work no. See if I have split the work across all the fellows equally then it is an optimal way of doing the best known sequential algorithm is  $n$  some complexity this the best known. So, I can only do as best as this if I do something better than that then that will seem to be best known correct right. So, if I say that each processor will do this much amount work if I take the sum of that total work is better than the best known sequential algorithm then I can make the same processor do one by one after this and I will get still better known better sequential algorithm are you getting this.

See the best known sequential algorithm say let you take 100 units of time right suppose there are 4 processors I have a parallel algorithm such that each processor can finish in 22 units of time. That means, what I will this 4 processor I will take one processor do the same task one after another then it will finish of in 88 unit of time this contradicts the fact that this is a best known sequential algorithm correct right. So, if I have the best known sequential algorithm then I have this many amount of processors and this time. So, this processor into time should be equal to the best it can be best equal to the best complexity of the best known sequential algorithm correct. So, that is that is the (Refer Time: 19:33). If it is not then it become a sub optimal algorithm this is a sub optimal whatever you have seen.

So, how to make it optimal? We will not have  $n$  processors right suppose I have one million numbers will you have one million course no. So, essentially I will have some  $p$  processors  $p$  is a constant sometime. So, what we will do? We will split up into  $n$  by  $p$  buckets. So,  $n/p$  let me call it as bucket 1 bucket 2 till bucket  $p$  each having  $n/p$  numbers ok. You understand this. Now I will do the prefix sum. So, I will give one processor to each one of them I will just go to sorry I will go to that.

(Refer Slide Time: 20:27)



So, totally there are  $n$  numbers. So, I will put it in to bucket  $B_1$   $B_2$  to  $B_p$ . So,  $n$  numbers,  $p$  processors, this is ideal situation right where  $n$  is very much larger than  $p$  now I will put the equal bucket. So, each will have  $n$  by  $p$  each.

So, I will do the prefix sum of each I will do the prefix sum of each how much time it will take and I will assign one processor I will assign  $p_1$   $p_2$  to  $p$  small  $p$ . So, I will assign one processor to each and ask them to do the prefix sum for each one of (Refer Time: 21:27) and they do it concurrently. So, how much time it will take  $n$  by  $p$  time at the end of this what will happen I will have one  $A_1$   $A_2$   $A_3$   $A_4$   $A_p$  right this is the prefix sum of the first  $n$  by  $p$  elements here the last this is the sum what will be the last after I finish the prefix sum. What will be the last element here? It will be the sum of all this elements here this will be the sum of all this  $A_2$  will be the sum of all this element,  $A_3$  will be  $A_4$   $A$  correct you understand this.

So, after I finish this prefix sum in each of this individual box the last element will be the sum of all the element in that block what is the final answer? If I add  $A_1$  to all this elements right then that is the final answer if I add  $A_1$  plus  $A_2$  to all the element here then it is a final answer, if I add  $A_1$  plus  $A_2$  plus  $A_3$  to all the element here that is a final answer, if I add  $A_1$  plus  $A_2$  plus  $A_3$  plus till  $A_{p-1}$  to all the elements here than

that is a final answer. So, what I should do now I have  $A_1 A_2 A_3 A_4$  till  $A_p$  I will

do the prefix sum of this then what I will get? I will get  $A_1 A_1$  plus  $A_2 A_1$  plus  $A_2$  plus  $A_3 A_1$  plus  $A_2$  plus  $A_3$  plus  $A_4$  and  $1 A_1 A_p$  minus 1 I will get this.

Now I will take this  $A_1$  and add it to all this fellows here I will take this answer and add it to all here, I will take this answer and add it to all here and so on I will take this answer and add it now I will get the final answer. So, how much time it required. So, I took  $n$  by  $p$  time already for doing this individual computation, then how much time will I take for I have  $p$  processors and I have  $p$  elements how much I will take  $\log p$  time to find this sum.

Then again I take one number and add it to every fellow how much time will you take  $n$  by  $p$  time right because I take one number each processor will take that number and add it to the list of this. This will take a one plus a 2 already computed and add it to this. So, they will do it concurrently again it is  $n$  by  $p$  time. So, this is nothing, but  $2 n$  by  $p$  plus  $\log p$  since  $n n$  is very much larger than  $p$  this  $\log p$  goes off. So, this is order  $n$  by  $p$ . Now number of processors is  $p$  now time is  $n$  by  $p$   $p$  into  $n$  by  $p$  is  $n$  and this is an optimal algorithm yes.

Student: Sir, prefix sum algorithm are we allow to carry on any index  $i$ ?

So, this is a recursive doubling you already talking (Refer Time: 24:56).

Student: (Refer Time: 24:56) like any given hide as to give the sum of all numbers from 0 to  $I$  right.

No, but you just have this  $p$  location and just execute recursive doubling that I tough earlier.

Student: No this will give a total sum that fine but.

No (Refer Time: 25:10) it will give the  $A_1 A_1$  plus  $A_2 A_1$  plus (Refer Time: 25:13).

Student: It give all that, but like.

After that the processor each processor will read exactly one of this location, so  $p_2$  will read  $A_1$  and add it to its element at that time  $p_3$  will read  $A_1$  plus  $A_2$  simultaneously and add it to (Refer Time: 25:30).

Student: Sir what we are asking is suppose there are ten elements and.

It is an array.

Student: It is an array and if I give the input as 3 it should give the sum from 0 1 2 3.

Why you need.

Student: Is that (Refer Time: 25:42) functionality prefix sum of adding all the numbers.

Adding all the number.

Student: Because in our previous recursive doubling since we are doing single element we can given a (Refer Time: 25:52).

No prefix sum is to create an array where it all sum of all prefixes.

Student: But we also solved a sub problem that given a index I we can give the sum from 0 to I 0 to I write.

Where did I saw?

Student: No like (Refer Time: 26:05).

Here also you can solve no.

Student: no here because we are grouping n by p. So, if x is like (Refer Time: 26:12).

After at the end you will get no.

Student: But like if the index is modular n by p only then it will work right else you will not get.

Why?

Student: (Refer Time: 26:23).

Why?

Student: Sir because.

Is not is not number 1 to  $n$  by  $p$  I can number it no this processor as to read processor  $p$  2 will read from  $n$  by  $p$  plus 1 to  $2n$  by  $p$ .

Student: (Refer Time: 26:37) if I want index something like 3 by 2 (Refer Time: 26:41) then I have to go to that index and then add  $A_1$  and so on so (Refer Time: 26:48).

Prefix sum is you will do for everything and query on this that is what we are being doing. Why should I say one. So, I want stop with I.

Student: (Refer Time: 27:02) no sir (Refer Time: 27:03) query on any prefix.

You set up you finish that own computation and then take from location and take the prefix sum, still you can do. So, if I am given 100 numbers I will do the sum for 100 and then query on element that is also possible here also, right.