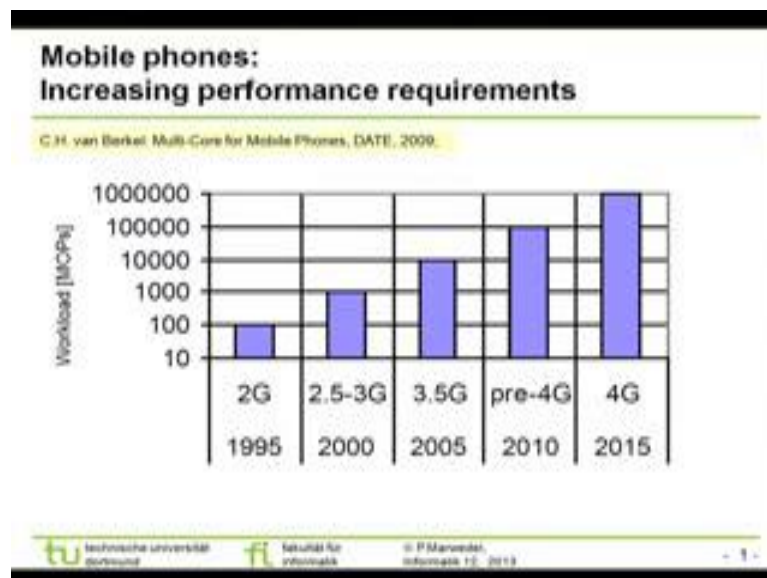


Embedded Systems Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 22
SD and DD Algorithm

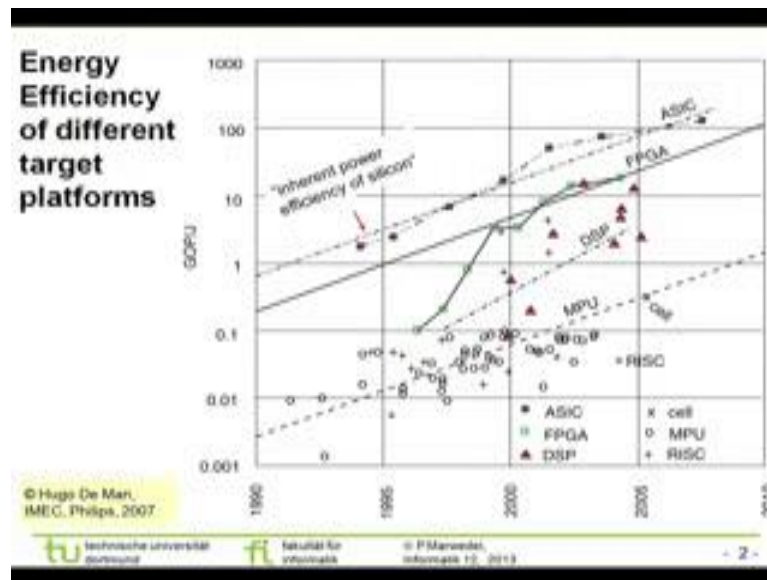
So, in the last class we were talking about dynamic voltage scheduling and we have not done the algorithms as yet, but even before doing the algorithms let us once again look at some examples which really makes this power management and power reduction issue very important.

(Refer Slide Time: 00:48)



Here is an example that we all of us use day to day that is the mobile phones and now with Jio and (Refer Time: 00:56) coming in everything is changing. So, you can see that the workload as we go from 2 G to 3 G, 4 G etcetera; the workload is increasing at this rate; workload in terms of mops or million operations per second. So, while it is actually increasing tenfold for each generation you can say, so more and more computational works has to be done. So, that is why; that is one reason which puts the challenge of reducing the power much more important.

(Refer Slide Time: 01:47)



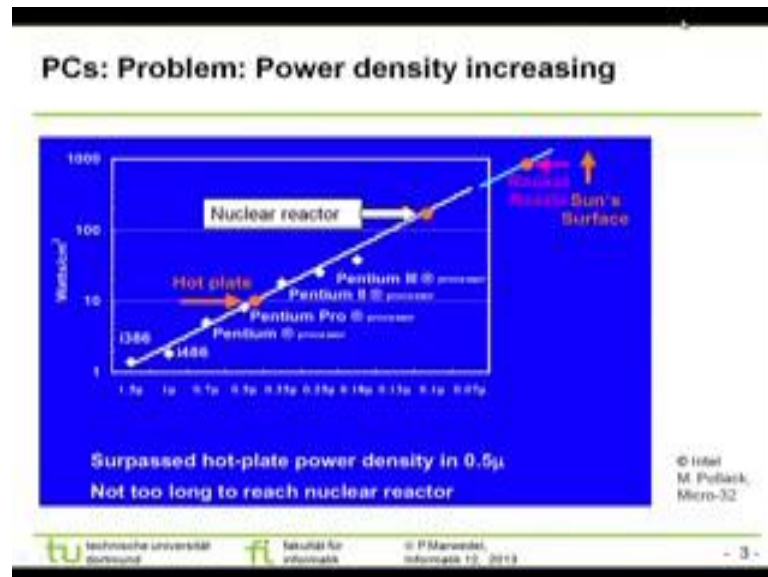
Now, also we can think of different target platforms, as you have seen we can move we can have a general purpose processor, we can have a DSP, we can have a FPGA's and we can have ASIC, so there is range of platforms on which you can carry out all embedded tasks. Now as you can see here as this diagram, this is a study by Imec Belgium that depending on different platforms, we have got the different GOP per joule; that means, the number of operations that we can carry out per joule.

In the earlier slide just now we have seen that how many operations we need right sorry here we have seen how many operations we need and as we are moving from 2 G to 3 G the number of required operations are increasing. So, here we see that the number of operations per joule per joule that can be carried out are varying and because of research on the x axis we are showing the time scale, you can see that we are being able to pack in more and more number of operations per joule.

Now, the winner in this case is of course, ASIC; you can see in ASIC we could pack in more or number of operations per joule where as the RISC architectures or the DSP architectures shown by these red triangles, they are all lagging behind FPGA's shown by these green squares are also lagging behind. So, the pursuit is therefore, to proceed, to try to reach do something on the DSP processors or do something on the microprocessors or FPGA. So, that we can reach somewhere here that is the happy island for us, we want to go to that level, so that is the pursuit. So, this is will give you some idea of you see this

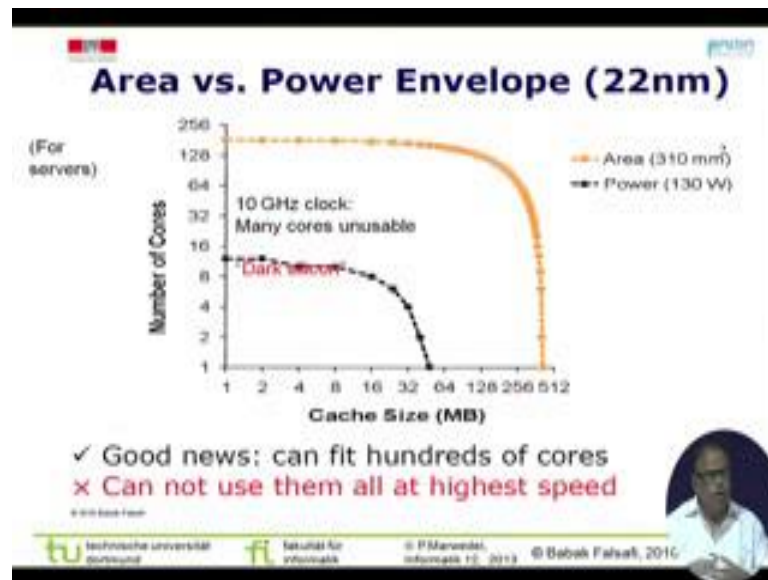
curve is showing the inherent power efficiency of silicon. I can actually try to follow this path as is being shown by this arrow, so our pursuit will be by in 2010 is already crossed, so we should try to be somewhere here.

(Refer Slide Time: 04:31)



Now, here you see the problem of the power density increasing here as you are moving to more dense packing in silicon; smaller microns as we go on then our on this side you can see this and on this side you can see the power expended per centimeter square. So, as the power goes on here if we go on packing more and more will be somewhere here with 0.1 micron which will be the heat generated in a nuclear reactor and if we go on further then may be ultimately here; the rocket nozzle or the suns surface. So, we cannot afford to do that therefore, right now we are somewhere here and that in an earlier lecture I had shown that is like an hot plate and you can cook an egg on that easily. So, that is what has been reached by pentium pro, so that is the scenario that we have already surpassed the hot plate power density, so what to do about that.

(Refer Slide Time: 06:01)



Now, let us look at this and try to understand this in a little detail. Now, there has been some benchmark organizations which have made some studies about the projected area that we can have and the projected power that we can dissipate and from there we could find that the area will be limited to around 310 square millimeter and the power will be limited to 130 watts because if we go more than that we will not be able to dissipate that. Therefore, we are having an envelope here on this side we have seen more; the number of cores.

If I can put in more and more number of cores the efficiency will increase; on the other side we can increase the cache that will also increase the performance the speed. So, one option can be that I can have large number of processors, each with small amount of cache that is somewhere here or we can go down and reduce the number of processors and increase the number of cache; whatever we do, we are restricted by this envelope because we are restricted by this area. So, this is somehow our possible space in which we can explore.

Student: Is the area limited?

The area is limited by technology, now here we can see that we can do it with 10 Gigahertz clock or something. Now as we go on increasing the number of cores say I go on this side, somewhere I have increased the number of cores and reduced the number of cache. The main problem that will come is the power, so there is another conflicting I

mean conflicting constraint on power. Therefore, I am having a zone which is defined by my power constraint beyond that I cannot dissipate. Therefore, although I may have for example, here 150 or 170 processors here, but I cannot actually power on at a time I cannot power on more than say 10 or 12 processors.

Therefore, you have got a huge amount of silicon but you are not being able to power on all segments of that, you are not being able to utilize that all the time. So, this area also you are not using; that is known nowadays by the term dark silicon; some part of the silicon that is never lighted; that means, we are not putting power into that.

Therefore there will be some segment of dark silicon, so beyond which we cannot I mean as we go on increasing the clock frequency, increasing the number of processors; the power goes on goes up. So, if I drive all the segments then the power consumption will shoot up, so that is known as dark silicon.

(Refer Slide Time: 09:51)



Now; obviously, you can see that although this was the dark silicon part because I am constrained by this black curve, I can say compared to this scenario; I can be anywhere in this zone by changing the supply voltage. We have seen that by voltage scaling, so that is I am once again establishing the necessity of voltage scaling. So, by doing that depending on different say as I go on with different volts, I can get different frequencies as you can see with 0.63 volt I go to 9 Gigahertz and get this curve with 0.21 volt and I go for 1 Gigahertz, so I get the blue curve something like this.

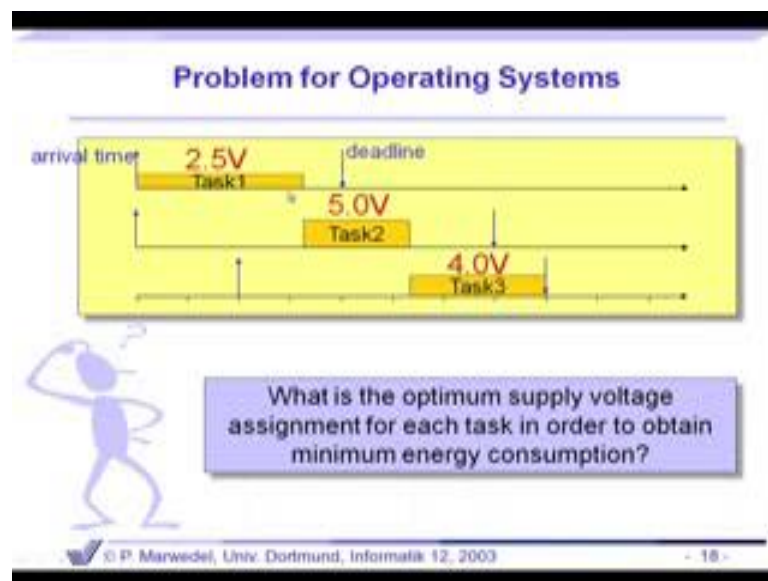
So, I can explore all these, so this is the zone which I can explore, I can utilize the (Refer Time: 11:00) to utilize the dark silicon; this is the dark silicon part because I was not being able to explore entire thing because it was restricted by power, but now I can change the voltage and go on expanding my zone where I can power on; however, in this study we are ignoring one thing that is the bandwidth; bandwidth means as I multiply the number of processors, the communication over it will also increase and that will tell on the speed and performance. Therefore although it is being shown like this typically it has been found that optimally we cannot have more than it is all predicted, we cannot have more than around somewhere here around 44 cores.

So, still if I put in pump in more and more cores, some of them will be unused as dark silicon. So, given this, so that establishes that given my power density and the performance requirement, I am now trying to activate more and more processes by doing voltage scaling which we have seen in the last lecture; so now.

Student: (Refer Time: 12:34) dark silicon.

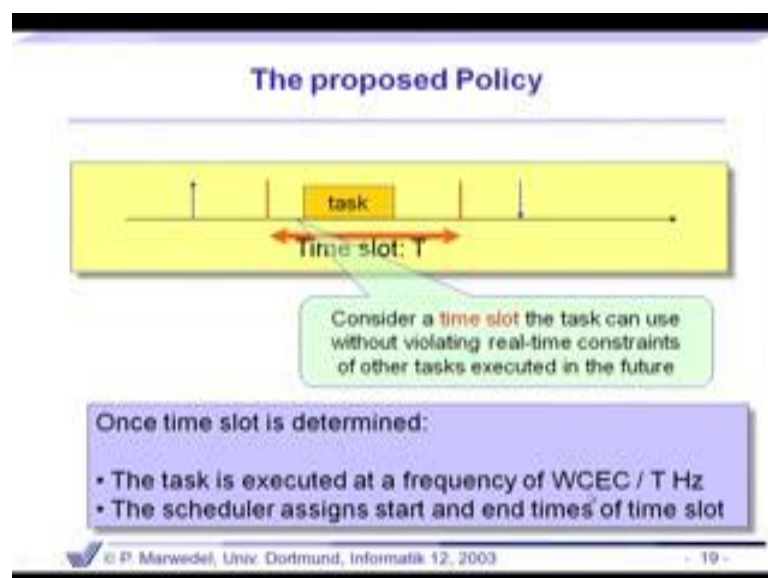
Dark silicon range was actually here top over there; dark silicon range is actually here this part because here is the power constraint as I go on you; see if I go on above this then my power is being violated; power constraint is being violated. So, I am not this is my dark silicon zone actually, so that is where; that is the part yes; I am trying to get that part activate that part by changing the voltage, but still there is a limit because there are other contradicting factors as well.

(Refer Slide Time: 13:28)



So, next in the last class; we had seen that we want to the operating system will take care of selecting the actual voltage that should be fed to a particular task. Now our task was to find out the optimum voltage by which we can run the different tasks; optimum voltage distribution by which; so for which task I will allocate which voltage that has to be decided by the operating system.

(Refer Slide Time: 14:04)



Now in that regard we had so we talked about the policy; basic policy was this that we will have a feasible time slot for the task; feasible time slot in which it can be allocated

without violating the other constraints then I try to run it, I allocate the frequency that voltage which is the worst case execution time by the frequency and therefore, I will get this task because this was the range, I reduced the voltage it will come to that; that is my objective. So, every time I will try to find out what is the feasible zone in which the task can be started so deadline is not violated, and other tasks deadlines are not violated and I will give the minimum voltage with respect to that.

So, there are four different algorithms we can have one is of course, which we are not discussing; one is known as the S S algorithm.

(Refer Slide Time: 15:11)

The diagram shows a flow from 'SD: Static scheduling & Dynamic Voltage Alloc.' to 'The arrival times of the tasks are known', which then leads to 'DD: Dynamic scheduling & Dynamic Voltage Alloc.'. Below this is a table comparing the two algorithms across three parameters: CPU Time Alloc., Start Time, and End Time Prediction.

	SD	DD
CPU Time Alloc.	off-line	on-line
Start Time	on-line	on-line
End Time Prediction	off-line	on-line

Which is the static scheduling and static voltage allocation, but we are not considering that we are considering two different varieties one is SD; that is static scheduling means static ordering and dynamic voltage allocation, another; that means, here when I say static scheduling, the arrival times of the tasks are known beforehand they are known apriori, but if the arrival time of the tasks are not known then we will come to a scenario where it is a dynamic ordering or dynamic scheduling and dynamic voltage allocation.

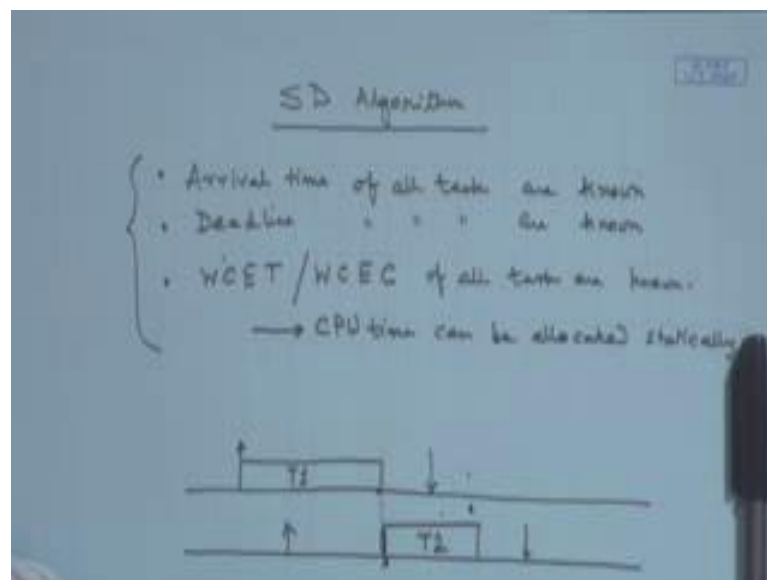
So if I draw this; I am looking at three parameters one is CPU time allocation, start time assignment; that means, when will the task start and end time prediction, end time will be predicted then for the two categories of SD and DD; the static scheduling and dynamic voltage, the CPU time allocation will be done offline, when I am giving allocating the

CPU because I know beforehand when the task will come. So before I start I can allocate the tasks, start time allocation will be online and end time prediction will be offline.

Similarly, here the CPU time allocation has to be done online because it is dynamic and all these things will have to be online. Now so we will first look at the static scheduling when I know; so what is the difference between these two; the difference is that in one; I am having I know apriori when the tasks are coming and in the other one I do not know when the tasks are coming.

So, let us start with the S D algorithm where we know in the S D algorithm.

(Refer Slide Time: 19:35)



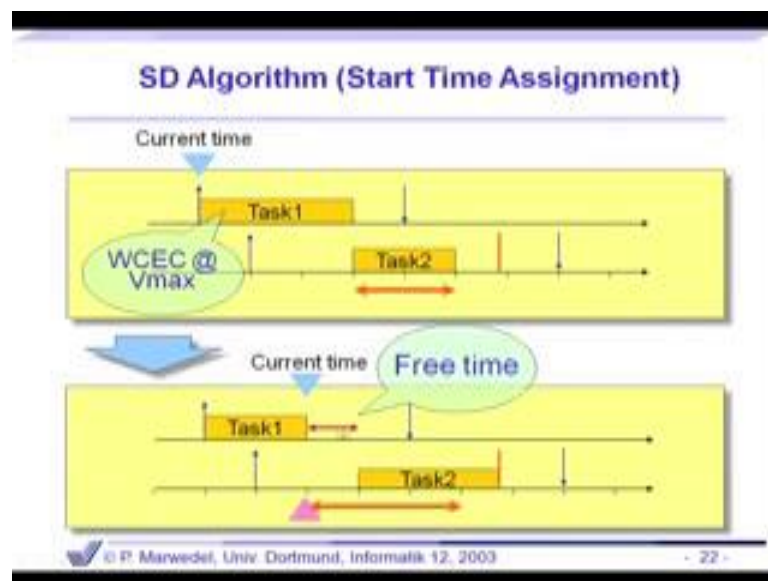
We know the arrival tasks of all the arrival time of all tasks are known, deadline of all the tasks are known; each task is coming with a deadline because mostly we are trying to study this embedded thing in terms of real time systems allocation, so deadline of all the tasks are known. Now this is an assumption that we are taking the worst case execution, we often say worst case execution time or we say worst case execution cycles; the number of cycles that are executed of all tasks are known.

So, this means that the CPU time can be allocated statically; since all these things are known I can allocate the CPU time to be allocated statically for example, let us draw something here; here is my timeline and say a task has been generated here; this is task one and its deadline is somewhere here alright; this is the arrival time, this is the deadline

and task two has arrived somewhere here, but I cannot schedule it here because I am assuming an uniprocessor scenario and its deadline is somewhere here and I can allocate the task immediately after task one ends and suppose this one is over. So, this is a scenario of static scheduling alright I have not talked about the voltage, I can do that.

Now, suppose I mean this task is run at a higher voltage then this will shrink, this will come down and then I can allocate this earlier. So, what is happening is CPU here; it is a typical case where I am being able to allocate the CPU time; I know that since I know the worst case execution time, I know when the CPU will be free. Therefore, I can allocate although it has come here, I can allocate it to you here and why is it online because the arrival the task has arrived and I know the worst case execution time of that. So, that is the basic assumption, but we have not; so we can find a schedule, but we have not found out as yet what would be the minimum energy scenario.

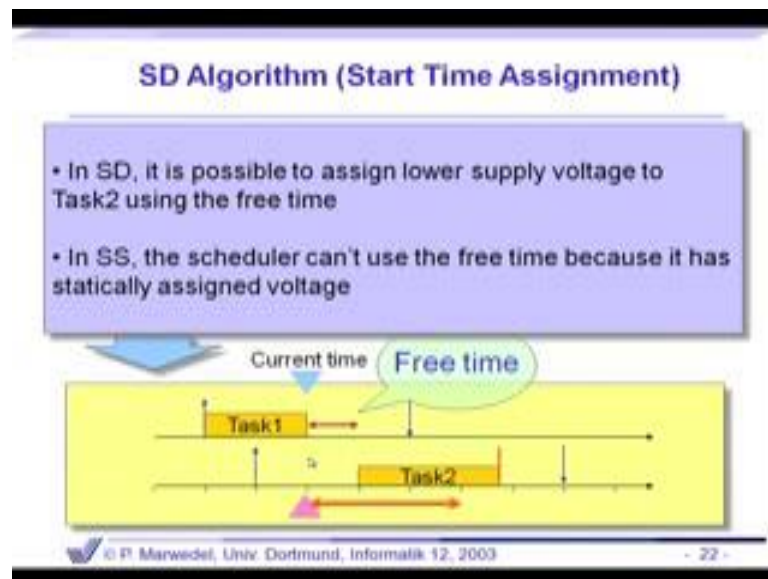
(Refer Slide Time: 23:34)



So, now let us look at this SD algorithm, so task one is here the diagram that I had drawn just now; task 1 is here task 2 is here. Now I can allocate this I know the worst case execution time and if I run it at V_{max} ; that is the max voltage then this task 1 has been finished earlier. So consequently what happens, I get some free time here because my statistical scheduling was here, my static scheduling of the task was at this point, but I can now dynamically by changing this voltage.

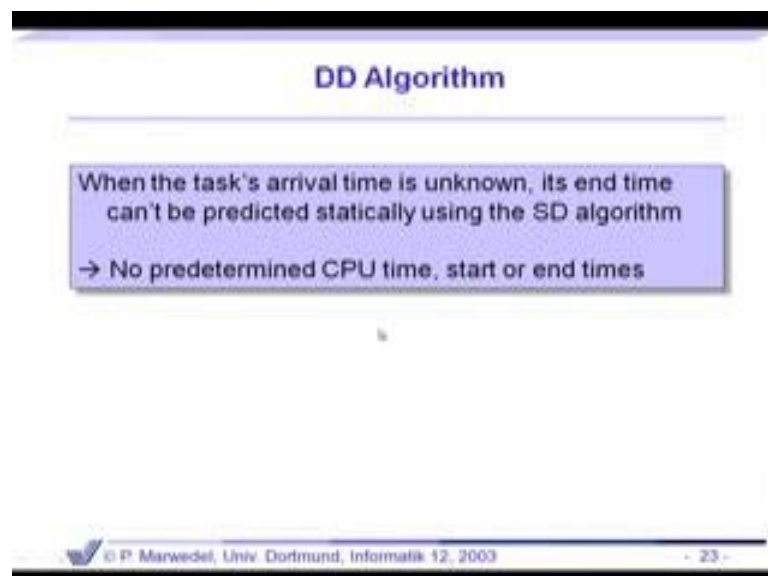
Now I can push it over here in this free time and as I push it over here; I get a longer zone in which I can manage; you recall that earlier we said the basic policy is we have to find out the feasible zone and then we will run it at the minimum possible voltage, so that it fits in that feasible zone does not go beyond that feasible zone. So, here what will happen; this task one has been run at a voltage and therefore, it has finished earlier.

(Refer Slide Time: 25:00)



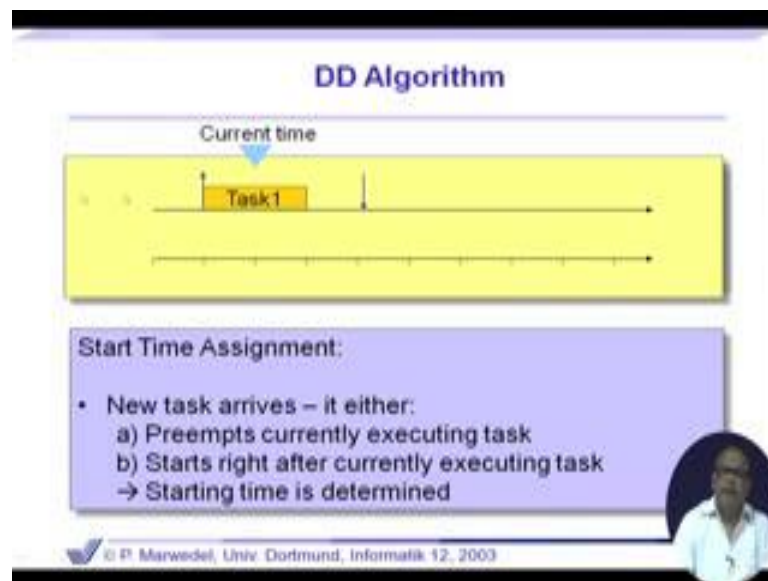
So, I can assign lower supply voltage to task 2 which I could not do in SS; if it was statically scheduled and static voltage assignment.

(Refer Slide Time: 25:16)



So, in this case I can certainly do that, so that is the scenario of SD algorithm static ordering but dynamic allocation. Next we come to the dynamic allocation and dynamic ordering both are dynamic that scenario. In this case as I said the tasks arrival time is not known and also its end time cannot be predicted statically. So, therefore I cannot apply the SD algorithm I do not have any predetermined CPU time start or end time; it can come any point of time.

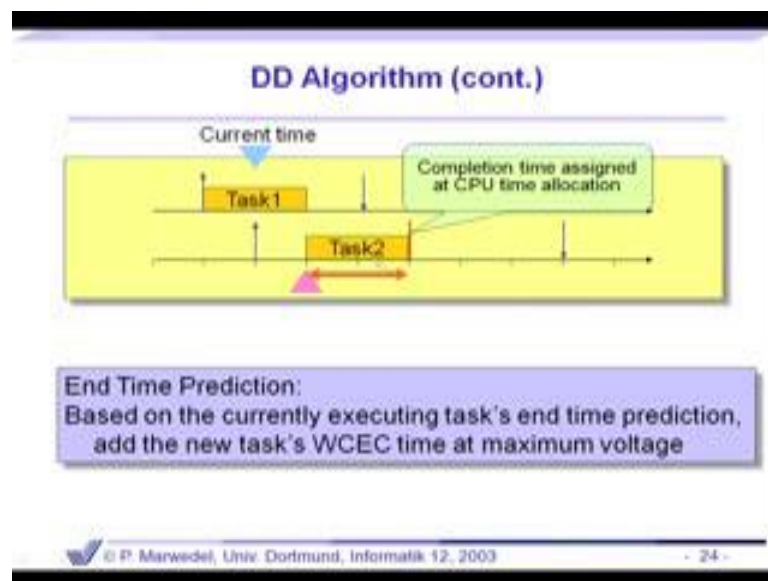
(Refer Slide Time: 26:04)



For example say; the current time is here and the task has arrived here presently where that the point will right now this is where we are, but some tasks arrive earlier, it has been allocated. Now new task arrives, now if at this point a new task arrives what can happen? Task 1 is already running and a new task arrives, the options are it can preempt the currently executing task, it can push it to the wait state and take control of the CPU, that is the preemption or it can wait if right after the currently this task, the currently executing task ends; it can be allocated so there will be some waiting time for this.

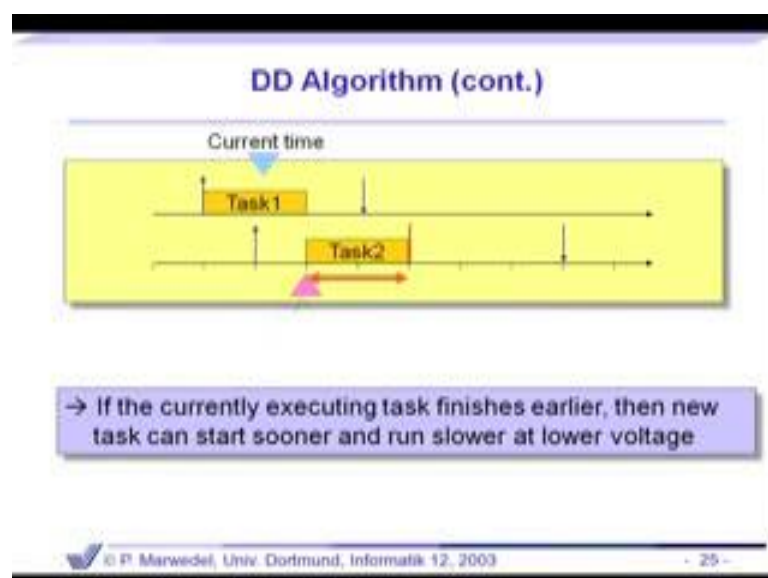
That depends on the priority of the task and the operating system scheduling policy that you have, you may like to preempt it or you may like to delay it a little bit.

(Refer Slide Time: 27:25)



So here we can allocate it after the completion of task 1 and still the completion time is allocated here and the end time prediction will be based on the currently executing tasks end time prediction because my start time is dependent on the task 2's, start time is dependent on task 1's end time. So, what can we do; I do not know that, so I have allocated, I have decided that I will allocate this task 2 only after task 1 ends. Now there is a deadline of task 2 also, so what would be my policy initially to start with? What voltage should I give to task 2? The maximum voltage, so that I ensure that the deadline is not violated. So, we add the new tasks worst case execution time at maximum volt.

(Refer Slide Time: 28:30)



So, the current time was here; now suppose this task now this arrived at this point task 2 arrived at this point, but I could not allocate it earlier; I allocated it here.

Now, if the currently executing task finishes earlier then I can run the task 2 earlier and reduce the voltage. So if task 1 ends earlier and this has arrived a little early so then I can spread it out over here; no issue, as soon as it finishes task 2 starts and thereby I could reduce the voltage to a large extent.

Student: (Refer Time: 29:26)

Now, you are dynamically doing that; you do not know apriori that is why it is a dynamic voltage allocation

Student: (Refer Time: 29:44) CPU.

Student: So, do you think (Refer Time: 29:49) time handling (Refer Time: 29:53) allocated earlier like it has to (Refer Time: 29:55) so this whatever voltage level is.

Allocated for.

Student: (Refer Time: 30:02) or this task 1 assigned voltage can also be changed task ones assigned voltage can also be changed.

Task 1s assign voltage can also be changed.

Suppose you see right now I am running at some voltage, now if I find an arrival which has given me enough scope then I can reduce task 1; say when a task has arrived, I find out let me go up here as the task has arrived I find its deadline that it is being met. So, I could very well reduce the task ones voltage because here I can find that I can meet the task 2s deadline. Even if I extend my time up to this alright, so ultimately by this but statically you cannot find a global optimum that is being a problem because you do not know when the tasks are arriving. So, you start with the rate and depending on that you will change your voltage allocation depending on the distribution of the task that has arrived.

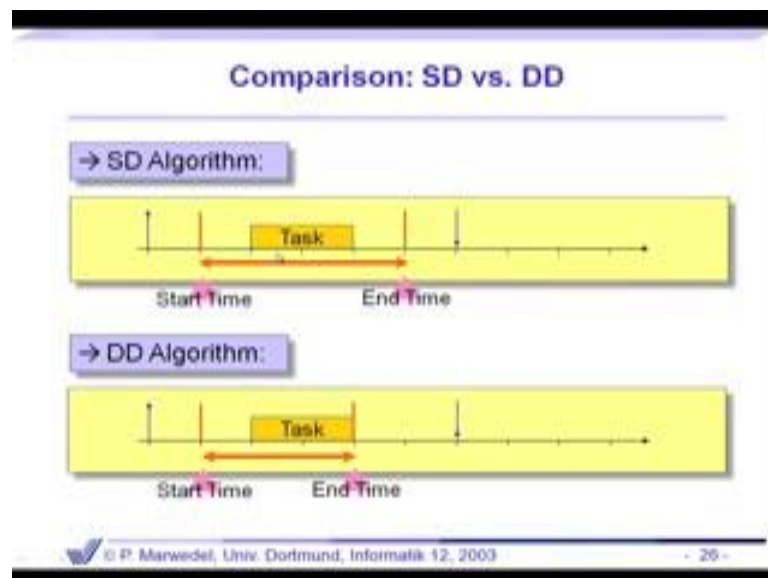
Student: Is that the deadline crossed the timing was based upon the deadline (Refer Time: 31:24).

This is coming from here this coming from here; the task 1 is right now running at this point, at particular voltage it is running here, so this point is that is where I can run the task 2. At the highest possible voltage and with this highest possible voltage, I can get the worst case execution time and this task will be finished by this; that is where we are getting this deadline.

Student: Do we need a (Refer Time: 31:51).

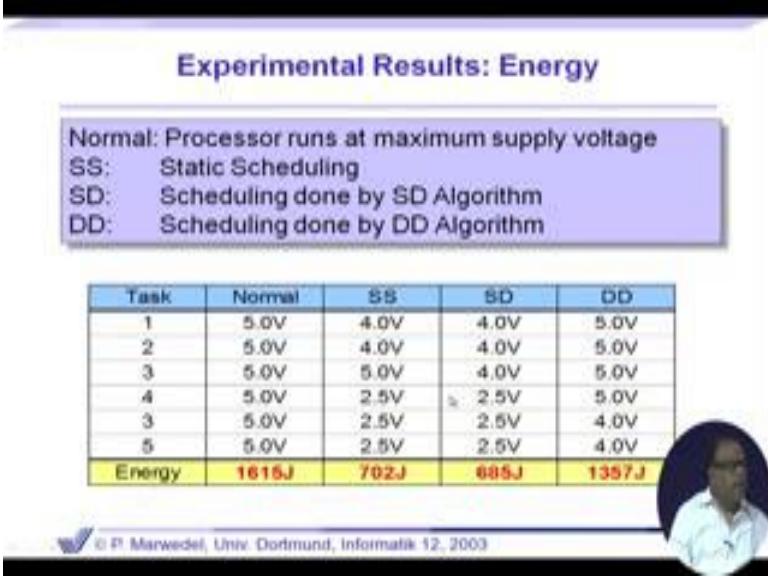
No we need not, if I run it at a lower voltage; so I could have suppose task 1 all the sudden decides. Suppose task 1 decides that I will be running at a lower voltage and it is pushed up here then task 2 will also be pushed beyond. Right now you can see only task 1 and task 2, but there will be task 3, task 4 also, so much opportunity may not be there, but in this case; obviously, there will be opportunity that task 2 can be pushed up to this because that is a deadline. So, this just an example case showing how the algorithm works, so here what is being shown? Instead of bringing in too much complicity; what is being shown the task 1 has finished earlier, it is finished earlier.

(Refer Slide Time: 32:47)



Therefore I can reduce the voltage of the task, it was arriving here, it has already arrived this one my queue, so I can give it that part. So, the comparison is here the SD algorithm I can minimize, I start with this zone where I know this zone, but in the dynamic allocation, I actually look at the end time because at best; I will be pushed up to this my end to worst case end time can be this one alright the deadline.

(Refer Slide Time: 33:16)



Experimental Results: Energy

Normal: Processor runs at maximum supply voltage
SS: Static Scheduling
SD: Scheduling done by SD Algorithm
DD: Scheduling done by DD Algorithm

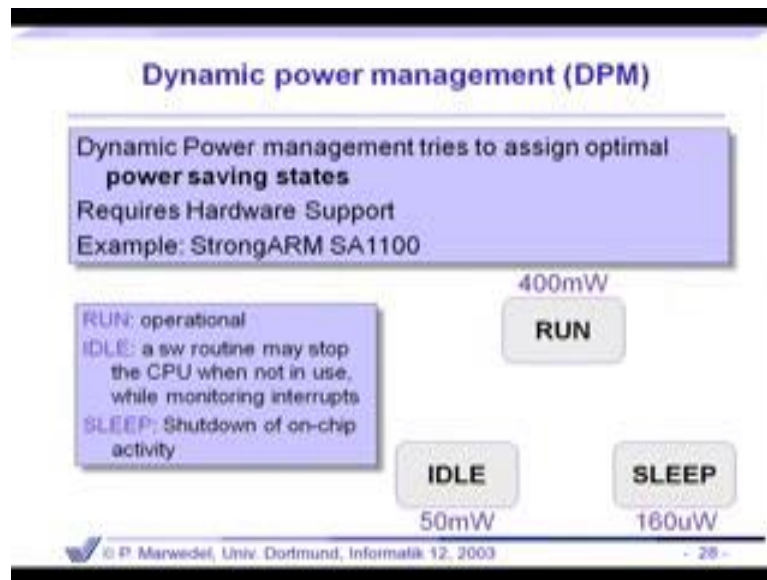
Task	Normal	SS	SD	DD
1	5.0V	4.0V	4.0V	5.0V
2	5.0V	4.0V	4.0V	5.0V
3	5.0V	5.0V	4.0V	5.0V
4	5.0V	2.5V	2.5V	5.0V
3	5.0V	2.5V	2.5V	4.0V
5	5.0V	2.5V	2.5V	4.0V
Energy	1615J	702J	685J	1357J

© P. Marwedel, Univ. Dortmund, Informatik 12, 2003

So, typically some experiments were carried out and the experimental results are interesting. You can see for a task normal means the processor is running at the maximum voltage; 5 volt all the time. Static scheduling we have done statically somehow looking at the deadlines and everything was known say 702 Joules are consumed.

If I do that by here I did not play with the voltage, but if I play with the voltage for the same set of tasks, we could get reduce the voltage up to 685 joules here, but what we did is here only I played some tweak, in the particular example that we have shown, whereas you can see the dynamic one is more than this, but is it fair to compare between these two? No because we have got complete information here, we do not have complete information here, but still by dynamic we can reduce the voltage as you can see compared to this.

(Refer Slide Time: 34:42)



So, that is the main advantage of this static and dynamic voltage. Next I will discuss about the dynamic power management scheme that we will do in the next lecture.

Student: What is the (Refer Time: 34:55).