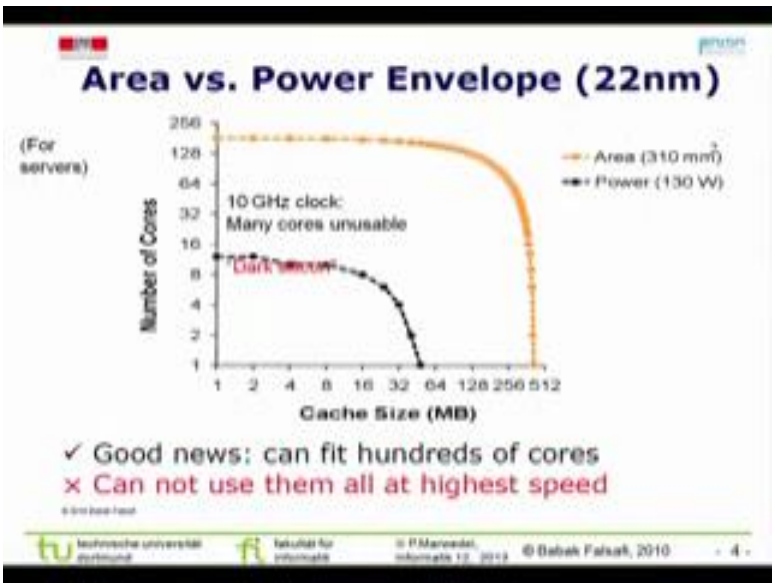
**Embedded Systems Design**
**Prof. Anupam Basu**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

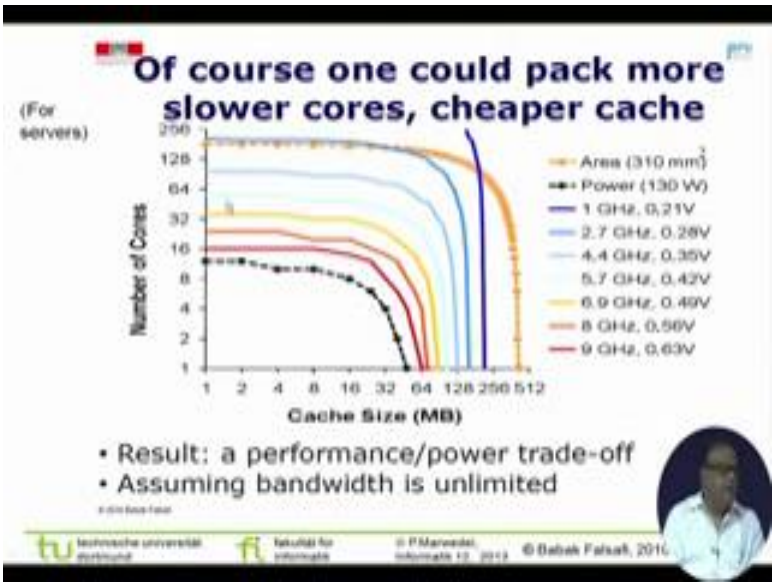
**Lecture - 23**
**Parallel Operations and VLIW**

(Refer Slide Time: 00:30)



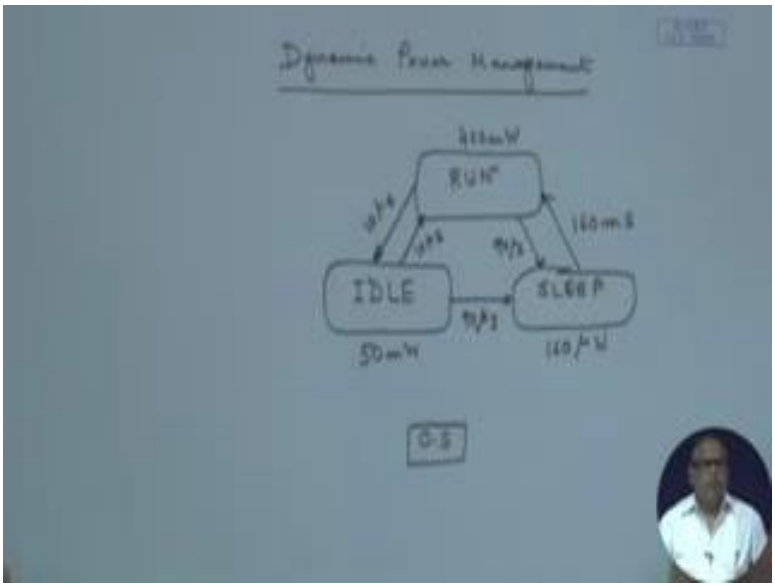

We were discussing about, Dynamic management of power, by how we can do that, by scheduling the voltage at different levels. And I think in the last class we were with this diagram where we had shown that there is a limit to which we can go with the area, with a tradeoff between the processors number of cores and the amount of cache memory. However, I cannot explore this entire space because of the requirement or the constraint of the power. So, because of power I will have to restrict myself say power has been predicted to be, should not be more than 130 watts which will be difficult to vent out. So, therefore, we will have to restrict ourselves to a particular zone of so much cache and so much processor. Therefore, many parts are not accessible; many space that was a possibility is not accessible.

(Refer Slide Time: 01:33)




And later we had shown that, even by voltage scaling we can go beyond this space and can explore this space, but because of the bandwidth issue, we will have to come to an optimal which is around 44 processors around this point. So, far we had done.

(Refer Slide Time: 02:20)




Next, what we will discuss today is about Dynamic Power Management, Dynamic Power Management. It works like this, there instead of a single processing state, that we usually have the operating system schedules it from the ready queue to the processing state etcetera you know. Now, that processing state can also be divided into a multiple tools

states. For example, there will be a RUN state, there will be an IDLE state, and there will be a SLEEP state. RUN state is normally operational, and let us assume that at RUN state it is consuming the power of 400 milli watt. Now, there are points when the C P U is not exactly running, but is waiting for some event to occur. So, till then the processor can remain idle and I can switch to the IDLE state and I said, I can switch to the IDLE state who am I here?

The operating system, the operating system will switch it to the IDLE state and say in IDLE state it will consume 50 milli watt power. And whenever there is a software routine or some event that calls for the processing, then there will be a wakeup from return from the IDLE state to the run state again. And this there is a typical delay of say 10 microsecond to go to the idle mode and 10 microsecond to return from this idle mode to this mode.

Here, the processor was not actually doing much, but was monitoring the interrupts and all those things, but there may be cases when we really do not do, we want to shut it down at that point of time, we can go from here to the sleep mode or also from here to the sleep mode. So, in the sleep mode it is a shutdown of the on chip activity, but the power is retained, its not the power off that you know, when our machine goes to hibernation the processing is not consuming any power, but the process the power is kept in the bank, I mean in the power supply. So, suppose that takes 90 microsecond, here also 90 microseconds.

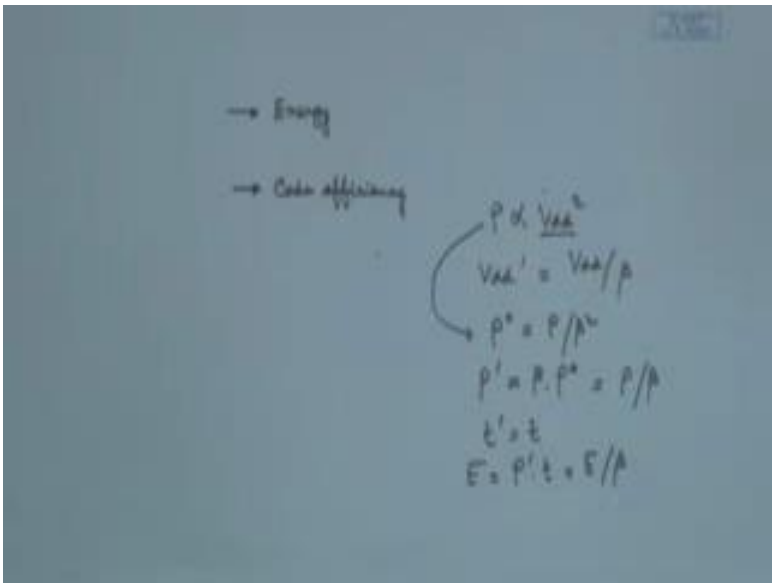
Now, when from sleep I again start working then I will have to return again to the run state and that takes around 160 millisecond typical. Say, so, if the operating system depending on the task scenario can switch between these states. I will be certainly in sleep, I am actually consuming 160 micro watt all right, much less where from is this microwatt coming I mean always you know, we have talked about that equation the power equation that we had talked about the energy that is equal to the alpha CLVDD square M.

Now, that was talking of the switching power, but as we are having more and more transistors, then there is a leakage current and leakage power actually is also not insignificant. So, those are being consumed anyway and the operating system will have to do this. Now, it is an open issue that how the it can be a nice thing to study as to what

would be the best algorithm to find out we have to predict because you see each of them each of these transitions are consuming some time therefore, if I often send it to the sleep mode then waking up from the sleep mode is taking 160 milliseconds, that may not always be affordable.

So, this is another way we can carry out the management of power this is known as Dynamic Power Management. There are issues in this as well where the algorithms which have been proposed I am not going into those algorithms in detail. Now, given this we have looked at the power aspect.

(Refer Slide Time: 08:13)

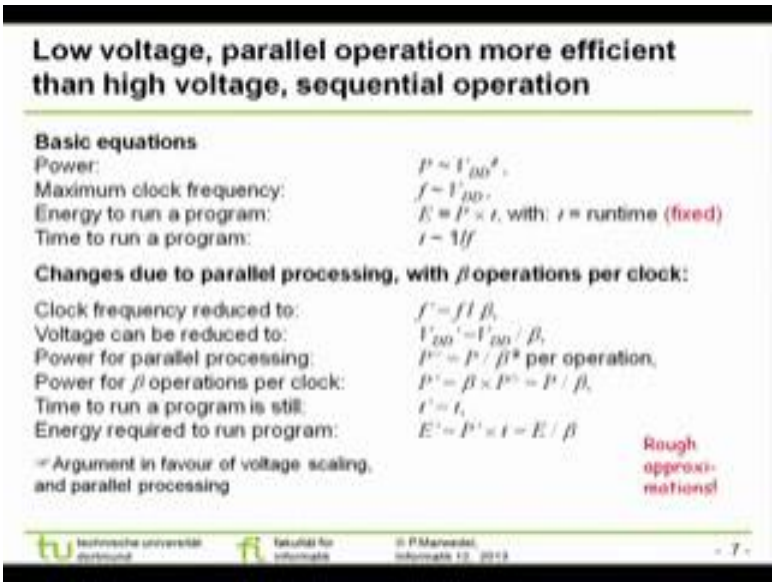



Now, the problems that we are interested in, we know that we want to minimize or optimize on energy, we want to optimize on may be code efficiency. Now, these are also related to each other whenever if I can make the code efficient if I take less amount of code space then; obviously, I will be running the program at an early time I mean, I will be completing it at an early time. Therefore, the total amount of energy that will be consumed is less.

Another point also you should think of; when our program is running, where are the power points, I mean where does the power get consumed? Is it only in the CPU? In CPU of course, whenever I am computing, depending on the number of cycles that operation takes power is consumed. Power is consumed also a heavy segment during memory access. Memory access, the more access to memory we do the more power we

consume. Therefore, another relevant thing if possible we will see later is that, how much can we reduce the access to the primary memory; if we can keep most of the data in the registers in the CPU or in the cache in that case probably these memory references can be reduced.

(Refer Slide Time: 10:09)




Next, what we will see now is another way of minimizing energy. Look at this slide. If I carry out parallel operations at low voltage, what was the problem? What was our problem when we did voltage scaling? Our problem was that we had some deadline, we had to meet that and for that we had to run it at a particular frequency and in order that we can run it at a particular frequency we had to give a particular voltage and as we give a higher voltage, we could reduce the time and with higher voltage we consumed quadratically more power.

Now, what if I now go towards parallel execution? Let us look here; our typical equations tell us that power is proportional to the supply volt, square of the supply voltage. The frequency is proportional to the supply voltage, the energy is equal to power times time, runtime is assumed to be fixed here and T is proportional to 1 F. Now, suppose, I had to do a particular number of operations now, I start doing parallel processing and instead of doing one instruction per clock or one operation per clock. I decide on doing beta operations per clock, I am doing beta operations per clock; then my effective clock frequency gets reduced to F prime, which is F by beta.

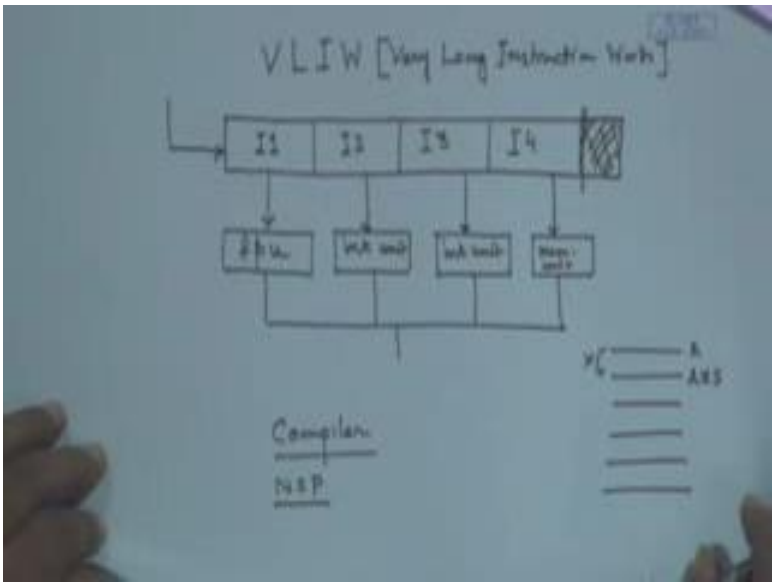
Similarly, the voltage is reduced to, I can therefore, since it is linearly dependent, think about the tau equation, tau was K times V DD by V DD by V threshold square. Therefore, ultimately it becomes proportional that frequency. So, if I reduce the V DD, as I reduce the frequency I can reduce the V DD, more delay I am allowing, then the power for parallel processing will be whatever P I was assuming here, whatever power I was getting here divided by V square per operation, because V DD has been reduced and the power is proportional to the V DD (Refer Time: 13:14). So, power for beta blocks will be this is. So, the power is being divided, power is proportional to V DD, V D square. Now, we are reducing V DD to V DD by some scale beta therefore, my reduced power. This is also if that be the reduced power that will be P by beta square and therefore, for all this is for one operation.

So, for all the operations, beta operations will be beta times this and that will be therefore, P by beta. Time is remaining the same, I cannot change the time, the time will have to do the same. Therefore, the energy requirement will be again this was P prime times T is equal to E by B beta.

Student: (Refer Time: 14:27).

No, what is the time? The time that is told over here is the time worst case execution time of the programs. I have to run it for so much time. Earlier, I was running it at a higher voltage, I am now running at a lower voltage; that is the only change I am doing. I am not shrinking the time here. So, that is this certainly if I look at the slide again, this certainly puts up an argument in favor of voltage scaling and parallel processing done together. I can do parallel processing and along with that I can do voltage scaling. So, that is another way that is another architectural way of dealing with power.

(Refer Slide Time: 15:29)




I will now talk about two other things, which some of you might know in your some advance architecture class that is very long instruction words; where instead of having one instruction I will have my instruction will have, a large number of instructions here. I am showing four instructions in one instruction word.

You know that the instructions are kept in the memory, and the program counter points to a particular memory location and that particular instruction is fetched decoded executed. Now, when it is being fetched, instead of fetching one instruction if I can have four instructions for example, taken at a time then what do I gain?

Student: (Refer Time: 16:50).

Number of, no, I am not taking from disks, I am taking them from the RAM. So, disk seeks are not reduced but.

Student: Memory.

But numbers of memory accesses are reduced. So, assume I have done some calculation mistake, it should be equal. So, forget about, suppose it is here, it is gone, it is nothing all. So, up to this is my instruction word now here suppose I have got my instruction one, here is instruction two, as here is instruction three, and here is instruction four; suppose I have taken four instructions. Now, there are further advantage is that suppose I, so therefore, I can now take the help of parallel processing because, suppose this one is

coming to a floating point, suppose this instruction is a floating point instruction, so it is coming to a floating point unit. This one is coming to an integer unit, this one may be another integer unit, integer unit means it is just doing the integer operation. So, I am assuming that my processing space is divided into identifiable and functionally independent components and here is something say; memory unit, memory access is being done here, some memory access.

So, therefore, what happens is I can, when I fetch one instruction and I can carry out four operations in parallel, I am decoding and all those things. Now, relatively we have seen that that will be consuming for decoding the combinational circuit will be run that consumes less power than memory access.

So, now how do we find out? Say; I am putting it four such independent instructions and clubbing them together, how do you know that when I have written a program like this? These are the 4 or 5 instructions or 6 instructions which of these can run independently. Say this; this instruction is reading A and this instruction is say taking A times 5 and putting it somewhere else therefore, these 2 are not independent. Therefore, I can put in this way only the instructions which are independent. The task of finding this parallely executable instructions is given to the compiler.

The compiler actually finds the parallelism. It is moved from the hardware, the task is moved from the hardware to the compiler and therefore, ideally this will avoid the energy, it will usage, unnecessary energy usage. But this left much to be desired, I mean there were lot of expectations which could not be directly fulfilled because, it was giving us low code efficiency for example, suppose I do not have a floating point operation here. In my, this code I do not have a floating point operation, but in my instruction sequence there is a floating point space. So, I have to fill up, I means again now here what is I?

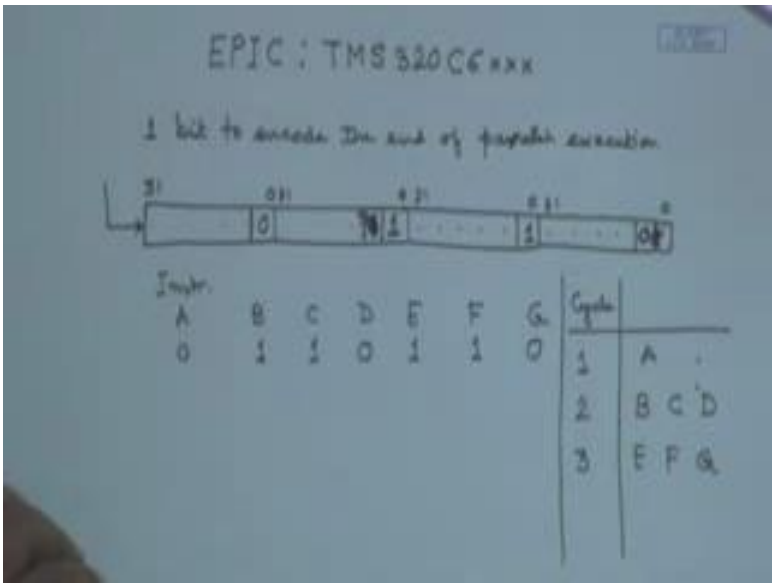
Student: (Refer Time: 21:17).

The compiler; the compiler will fill it; will have to fill it up this part as a N O P. So, this will have to fill up with a N O P, No Operation and therefore, often lot of space is wasted and we get low code efficiency in terms of the number of I mean, I want to reduce the size of the code. So, that also always does not happen to that extent, but to some extent it certainly pays off and this is another architectural move to reduce the power.

Student: Are these slots pretty fixed to the ordering given at (Refer Time: 22:03).

By the instruction set, yes, the instruction set has got certain floating point, some structure and we have to feed it up, will have to put the instruction set.

(Refer Slide Time: 22:23)



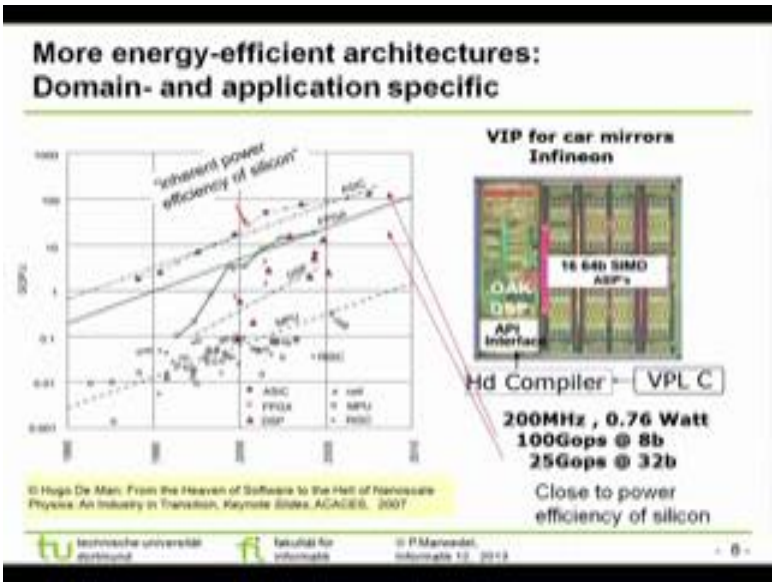

Now, one very popular; this was, this is being used. EPIC stands for Explicitly Parallel Instruction Code which was used by TMS as very popular DSP chip 320 C 6 and C 6 series was there what it does is very interesting. It does, it keeps 1 bit to encode the end of parallel execution. Like say; I have got 32, I mean each of these I have got 32 bits each field is having 0 to 31 and here again like that I have. So and here is the thirty first bit of this the zeroth bit of this like that, thirty first bit zeroth bit thirty first bit zeroth bit.

Now, the L S B of each of the instruction words, this is L S B of each instruction word is either 0 or 1. I am doing it wrong way, I hope you understand. This is not the, L S B is always being kept at 0 or 1 suppose, I make it 0. Now, if it is a 0; that means, the next instruction will not be executed in parallel. Suppose, now I have got instructions A B C D E F G. At the end of instruction A I put a 0, at the end of instruction B I put a 1, at the end of instruction C I put a 1, at the end of instruction D I put a 0, at the end of instruction E I put a 1, F I put a 1, and G I put a 0. These are the L S B S of each instruction.

Then how will the, for the different cycles, if I draw here the cycle versus the execution of the instructions. In cycle 1 what will be executed? A will be executed and because, A is not being executed in parallel with this. This one is not being executed in parallel, but when I come to B in the cycle 2, B says that let C be executed in parallel with me, C will be executed in parallel, C says let D be executed in parallel with me, D says no cycle 3 starts with E, E will be executed, and E says that let F be executed in parallel with D, F says let G be executed in parallel with me, and G says no.

So, you see EPIC is a little modification of or extension of VLIW and that is how that the embedded system designers or computer designers, if you ever work on some chip design or instruction set design. This shows how innovative people can become to design new types of architecture. So, architectures of a computer system is nothing but, mostly the instruction set design that you do. So, this also helps in parallelism, for reducing the time as well as, it is saving in the memory access by allowing us to fetch more and more instructions at a time.

(Refer Slide Time: 27:55)




So, next let us see here, we will try to show in the last class we had seen that.

Student: (Refer Time: 28:04) instruction is like the example which you gave was the eight instructions in the last section can it have.

I have not drawn all those things here. I did not draw all the instructions, I had just shown. I had drawn only 4 here, here I had drawn only 4, but actually these 8 should be drawn here.

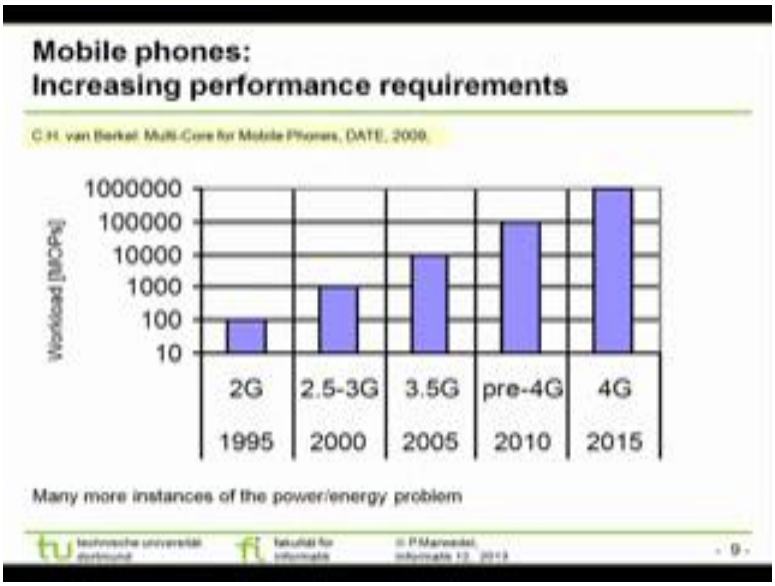
Student: This is the (Refer Time: 28:29) is it that.

That depends on the architectures. How many? How wide is your instruction word? Accordingly, you can predicate that this is known as see this 0 and 1 that I have put in as a single bit is a sort of predicated thing. We will soon see the arm; how we can compress the instructions in the much better way.

Student: in the last instruction can it have the last bit as 1 (Refer Time: 29:05).

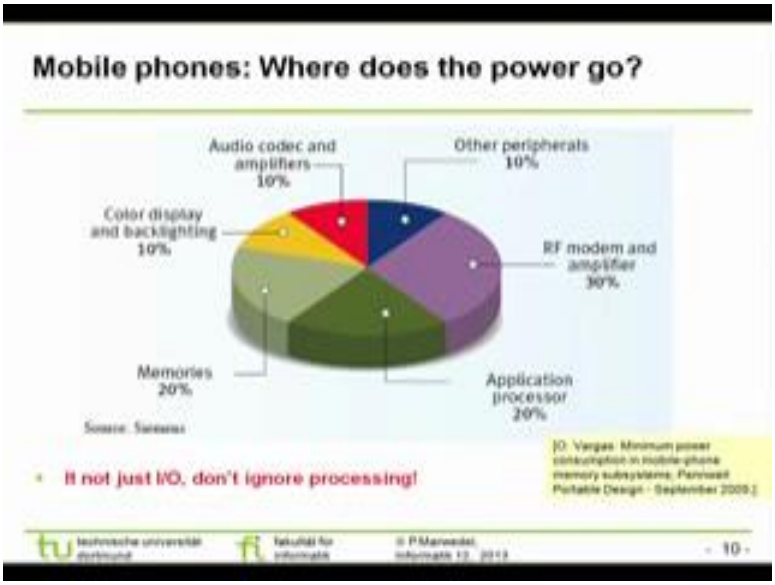
What you over to that mean? 1 means; see 1 means the last instruction. The next instruction should be executed in parallel. Now, for the last instruction there is no next instruction, therefore, that 1 would not mean anything. So, here you see, as we said that our happy point is somewhere here; happy island, I say that; we want to reach somewhere near the efficiency of Asics as regards the number of operations per joule, the power density. So, we can try to do that by different means by changing the compiler, by having some other code tricks. Normally, we can proceed towards; we are approaching towards this close to power efficiency of silicon. The power efficiency of silicon is this curve.

(Refer Slide Time: 30:03)



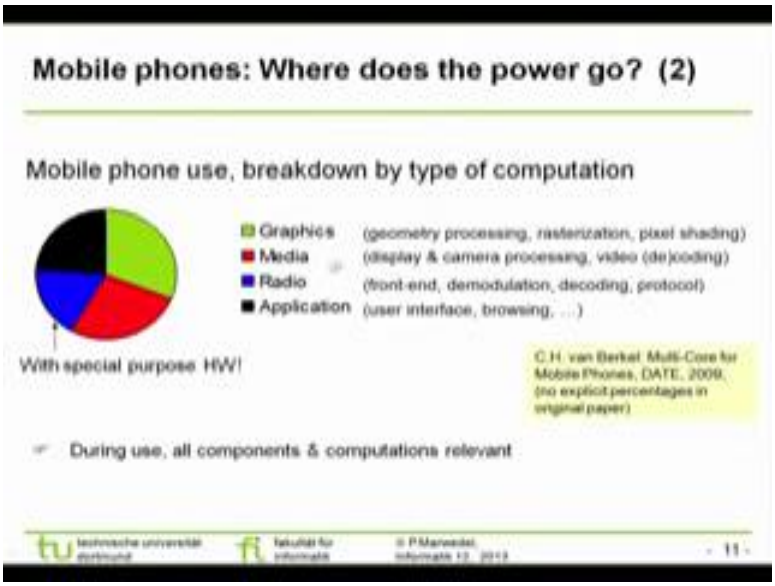
So, this one we had discussed in the last class, that tenfold increase of power requirement for mobile phones generations, after generation.

(Refer Slide Time: 30:16)



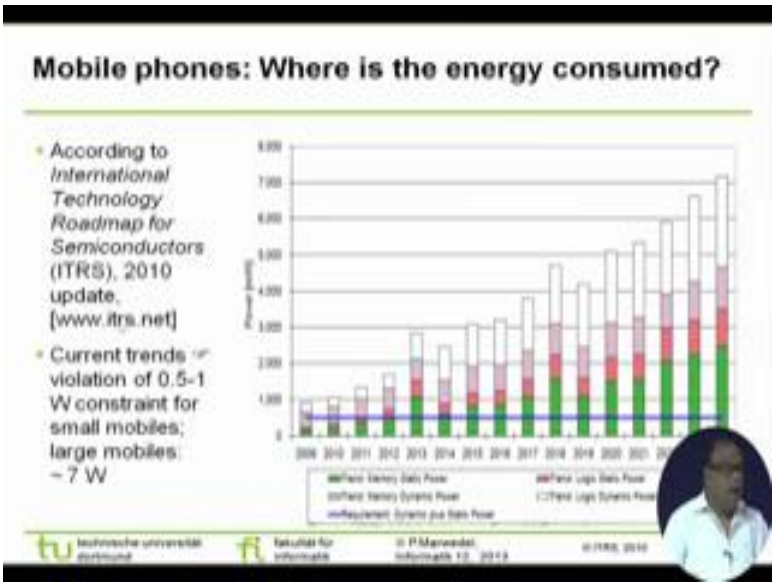

Now, let us look at, where does the power go in a particular mobile phone? Is it only in processing? Where is it? It is this pie chart, shows you that around 30 percent power is consumed in the modem and the amplifier, application processors are taking 20 percent, memory takes 20 percent, color display and backlighting that we use 10 percent, audio codec and amplifier 10 percent, other peripherals are taking 10 percent. Therefore, although, R F is a major consuming thing and we all of us know that, that is taking more power, but others are also not less guilty. They are also taking power, it is not that only this R F and networking is taking power; your color, smartphones styles, and all those things are consuming power.

(Refer Slide Time: 31:17)




Now, if we look at it in another way we can see that the graphics is taking some power; media is taking some power, radio and the other applications like browsing and all those. Here also these things we can, so, for each of them we need powerful processing. Now all of them, if I just make it in a single chip doing everything, single processor, core, one core, doing everything, then it will be difficult for me to manage the power of the individual segments. Sometimes may be you are not running audio, sometimes you are not playing the media may be, then certainly what we learn is that, according to whatever we have learnt till now, that we can put the power off in that processor. Individually, I can do that Dynamic Power Management for that processor which I could not do if that was a monolith, if that was a single processor.

(Refer Slide Time: 32:30)



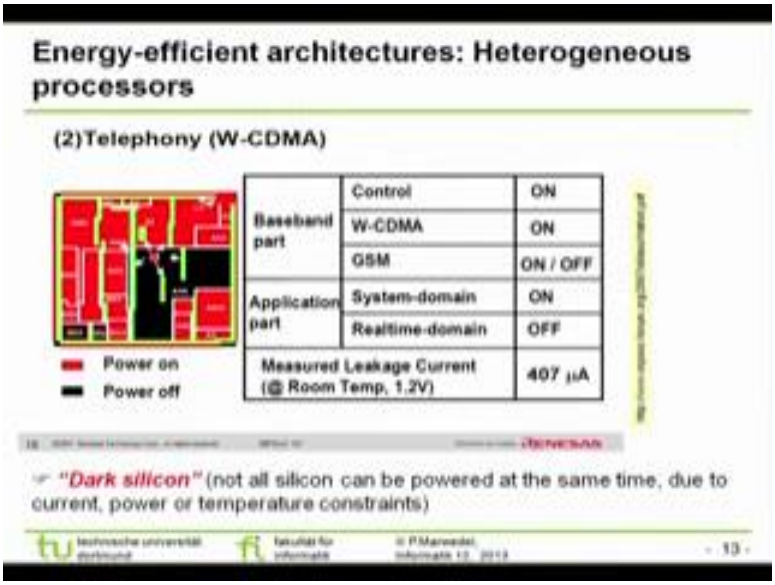

So, therefore, here also it is the same thing being shown and it is the data from the International Technology Road Map for Semiconductors in 2010. The scenario is not very good because you see, memory Static Power is shown by the green and Static Power means the leakage current.

Static power is the leakage current. Dynamic Power is our equation, that thing, switching current. Now, you see the static power, as you have pushing in more and more and more and more transistors, the static power is shooting up by 20-24 it will be where are we now, we are now somewhere here and this is the ideal that we can handle, I mean, so much power we can handle. We are already above that. Now, Logic State Power; let us see in stein 2016, takes a certain extent. Memory Dynamic Power means what? What does memory Dynamic power mean? The switching in the memory, I am accessing the memory and getting it is data that is taking up a considerable amount of power and the Logic Dynamic Power; Logic Dynamic power means? In the circuit, in the processor where reverse switching take place, all of them have to be reduced, so that the entire thing can be pushed, around this blue line.

So, we are actually, this is in milliwatt. So, the prescribed wattage is 0.5 to 1 watt and all of us for small mobiles, we are violating that for large mobiles, we are violating that heavily. So, more and more we go for the smartphones and this we are suffering from

that. We have to bring it somehow over here. So, what are the measures that people can take?

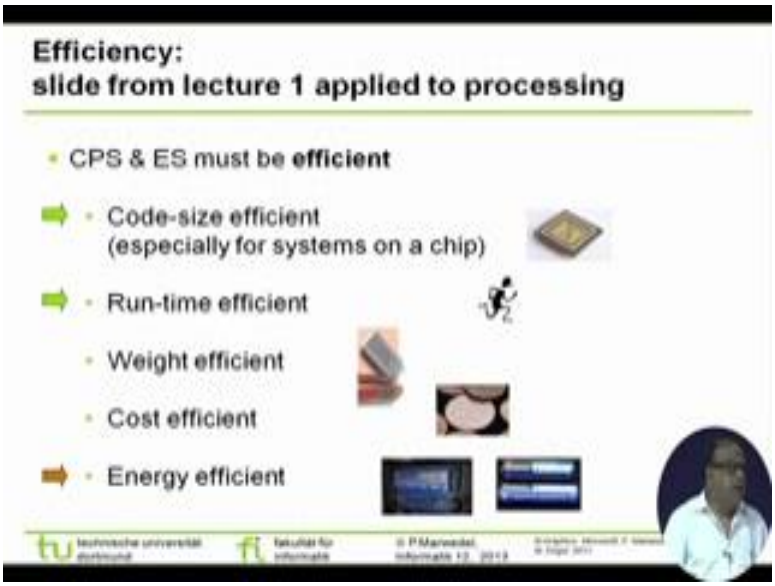
(Refer Slide Time: 34:34)




Student: Why does (Refer Time: 34:38).

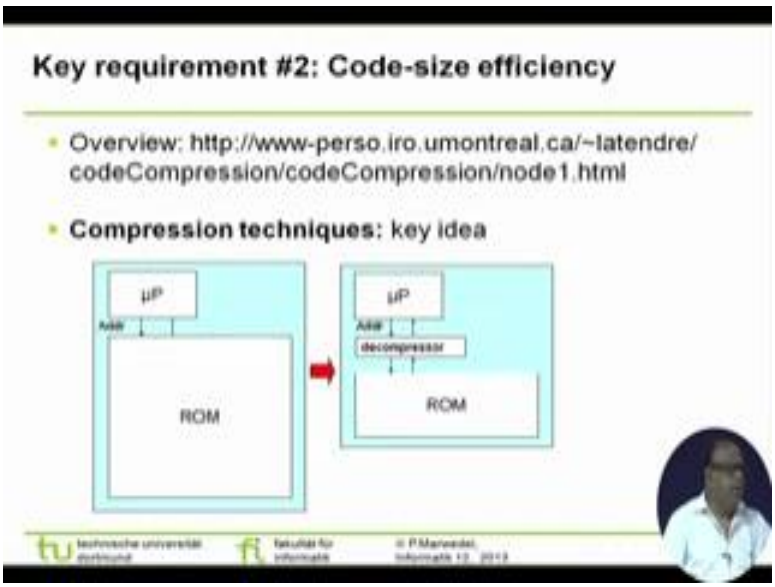
Now, you see here the processor, we are showing the red parts are the ones which are right now consuming power. And some parts, say this application part may be, I am not running by our DPM strategy Dynamic Power Management strategy, I can shut it off. So, thereby, we can have some part, baseband part is on. If it is a CDMA broad GSM both, say I can put CDMA on this can be on or off. I can put the real time task awareness not doing, I want to put them off and thereby, I can measure the leakage current etcetera. So, dark silicon you can see this.

(Refer Slide Time: 35:38)




So, now, we have talked about different means. Now, the cyber physical systems or the embedded systems in order to be efficient, there are so many parameters. We are not going to talk about the weight efficiency or the cost efficiency that is a different aspect although important. We are talking about the code size efficiency because, you see the code size efficiency is also important for the for the system on the chip, I do not have, I cannot have enough, I mean as much memory as I like runtime efficiency for the speed and energy efficiency. So, these are the things.

(Refer Slide Time: 36:26)



So, I will be talking about this, the way we can handle Code Size Efficiency. We have shown some code size efficiency or the VLIW was actually doing reducing the accessing and increasing the parallelism and pushing the overhead to the compiler. So, the next class we will be talking about the Code Size Efficiency. We will start with that the next class, but till now, let me quickly summarize what we have learnt.

Now, we have got two conflicting constraints; one is the performance, another is the power and the way, the methods that we have seen are the voltage scaling. The other one is Dynamic Power Management; we put certain things off. The other thing that we have seen today is also if we have got large instruction words, if we can reduce the number of memory fetches, memory accesses, writing into the memory, reading from the memory, then also we reduce power. So, we will take up the Code Size Efficiency from the next class.