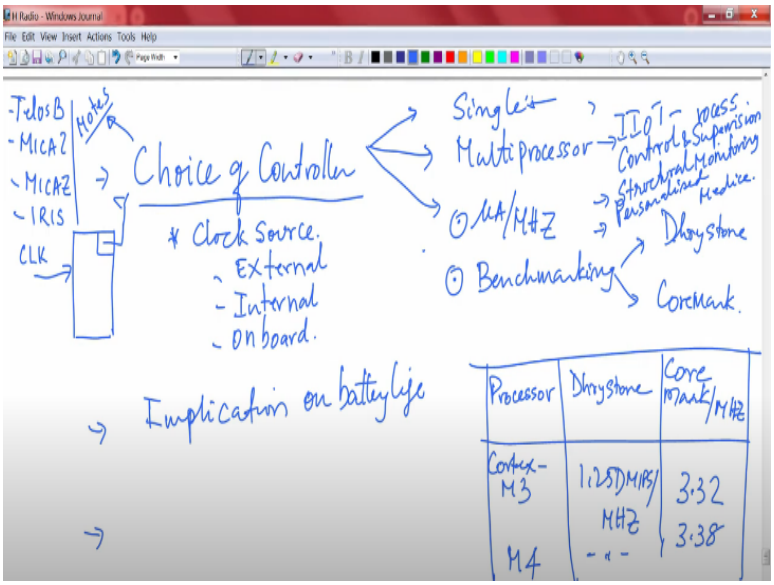
Design for Internet of Things
Prof. T. V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bengaluru

Lecture - 16
Power Management Systems - 01

(Refer Slide Time: 00:30)



So, let us begin again. I wanted to draw your attention to back to this choice of controller where I have clearly articulated that when you talk about a simple sense and send kind of application you can be talking about a single processor. Whereas if you are looking at much more serious applications; then those applications essentially will require multiple processors okay. Now the point is that often you will be wondering, why these, what are these applications which can be highly demanding?

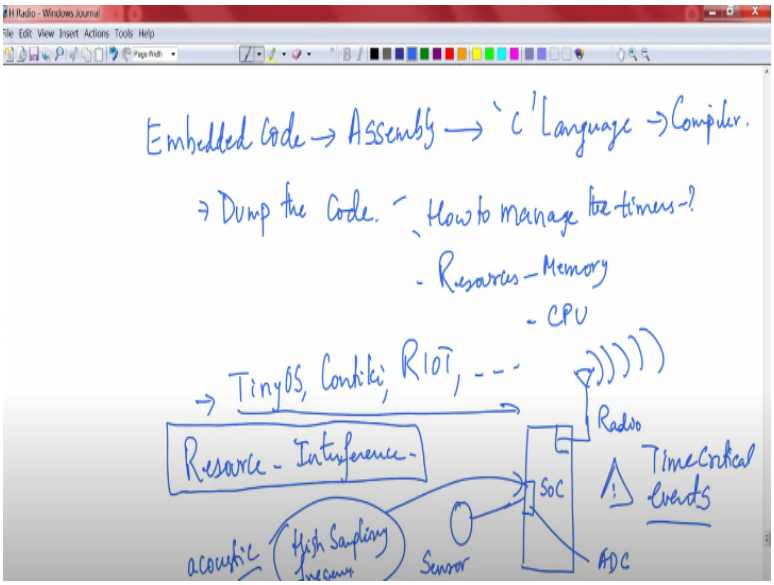
Now one of the things that we can think about is industrial IoT. You need this for industrial IoT application. So, I will call it IIoT applications. So, that is one thing. Anything that has societal and economic importance such as industrial IoT; when you talk about that we will talk about the industrial process control and you talk about process control and also supervision then environmental and structural monitoring, very demanding applications.

Then personalized medicine, you can go on like this folks. Several examples can be given. Then home automation, traffic control and so on. So, all these do not really come under the sense and send applications anymore. So, that is the; key difference in sense and send you are just doing extremely simple activities, quite trivial in that sense and they are extremely low power. They can work for the required 10 years of lifetime and so on.

But those applications are one type. So, if you have to choose a controller or going away from the simple sense and send kinds of applications, actually for sense-and-send we can talk about some controllers and I will write down a few. Telos B, for instance, you might have seen Telos B then you might have seen its predecessor MICA 2 then even its predecessor is MICA Z, then IRIS. All these are mote class devices.

They used to be called motes. Simple motes, 16 megahertz kind of processors, in fact, MSP 430 a very well known processor is actually used in Telos B and it is used for those simple applications. Those are one type of applications but then they are not the kind of ones that can be applied here and I will have to motivate ourselves. Why do you need a deep multiprocessor and what are the limitations of these existing ones?

(Refer Slide Time: 03:57)



Let me draw your attention to the earlier, very quite earlier years. See, writing an embedded code used to be hard. It moved away from embedded code in terms of writing assembly language, assembly code. It moved away to writing it in some sort of a C language. You write it in C. And then you can use a compiler. Isn't it? So, that you are comfortable writing in C code? So, this has been something you can do and then dump that code.

So, you could simply dump that code on the flash. Dump the code. Another way is you can do it this way. But then if you have to maintain timers it used to be very hustles. You have to manage the timers yourself and you have to write the code managing timer. So, I will say dump the code. How to manage the timers, how to manage the other resources, memory for instance? And also CPU all this you have to manage yourself that means as a programmer you have to keep track of everything.

So, people went ahead and said this is going to be very hard. Let us start developing using an operating system. So, people came up with TinyOS, Contiki then real time OS, I think it is called RIOTS. It used to be called RIOT. So, many operating systems were there. So, the operating systems job would be to manage all these timers and other resources and all that and provide an abstraction to the programmer. But to be honest this is very difficult for any changes that you want to do.

For instance, even the people who develop these compilers for these operating systems to run on these mighty modes, I mean, small modes have not been very successful in providing a foolproof way of ensuring that performance is sort of extracted to the maximum. So, that has been the problem. For example, developing software for heavily resourced hardware platforms is quite labor intensive and the coupling of the functional and non functional properties are quite hard.

So, I will tell you why this operating system based systems, operating system based embedded systems where it is very hard for us to manage. Let us take an example. Otherwise we will never be able to get to that. There is one problem for this kind of single processor based approach

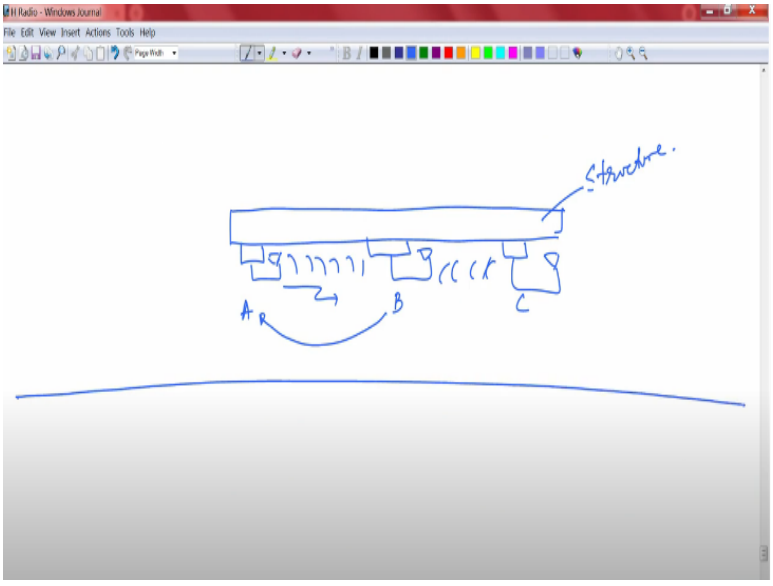
including an operating system particularly with respect to resources. In fact, you have resource interference.

Think about the following folks, think about a single controller which has an ADC and it is connected to our standards. This is a picture that we know very well. This is the ADC and this is the well known radio and all of it is part of the SoC. This is ADC here. If you look at this picture closely you will realize that supposing this is high sampling frequency you need very high sampling frequency here. It could be let us say, acoustic, microphone.

If you are using an acoustic sensor like a microphone you will be sampling at very high frequency. And if you are using these acoustic sensors, let us say, what we described as previously the issue related to structural monitoring. Now I am trying to retrofit a single processor for structural monitoring. So, let us see what actually happens. So, I am using it for structural monitoring.

Now the issue will be the following. Because you have a single processing resource MCU needs to handle let us say some time critical events. What are those? Folks; now comes, the story.

(Refer Slide Time: 09:51)



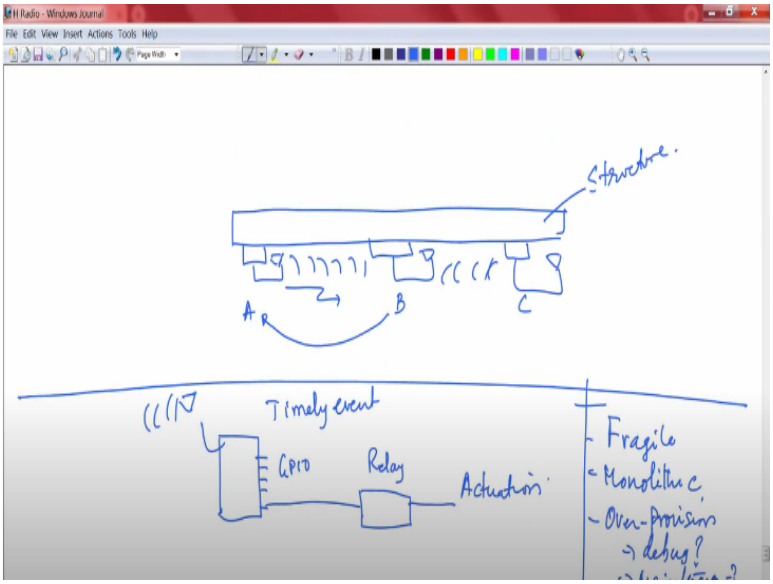
This is your bridge or some structure that you want to monitor and you have put embedded sensors at different places. Depending on how long or how big your structure is you would put embedded sensors and they are all communicating. Is not it? They are all having communication units. I just show them by some radio. This is your structure. Now see the problem. Here in this picture this guy is doing high sampling frequency, this SoC ADC.

At the same time there are signals coming from this one. This state is A, B and C. There are three nodes here. B is giving some data to A about something. Maybe it has sensed some anomaly and it is passing that information to this A. A has to respond in time. It is a time critical operation it has to respond in a given deadline. So, it becomes time critical. It is almost impossible folks, even if you have the best OS running to work on this single processor environment.

Because here what happens? This time critical event in this case is coming from where? It is coming from the radio. So, the same processor should it look at the radio or should it look at the high frequencies and sampling that you have connected, the acoustic sensor? Both are conflicting requirements. Therefore, what will happen? CPU resources will get into interference. So, this is a problem which cannot be easily solved if you try to look at the single processor situation.

You can also take another example. Let us take another example to convince you. Let us take one more example. Example 2.

(Refer Slide Time: 12:31)



In the second example you suppose that the application requires a node to react timely to events. For example, you have a sensor node, then you have several let us say GPIO pins and to one of the GPIO pins - you have connected a relay. And this relay is supposed to do actuation and this is a timely event. It has to happen within a certain time. It has to quickly perform an actuation and so again you have to receive the data over wireless and within certain latency within a certain bounded delay you have to issue the actuation to the relay.

The relay command has to be given so that it can get actuated in quick time within the required time. These kinds of things are not going to work even if you talk about a single processor with an operating system. So, you can summarize and say with an OS like TinyOS or Contiki and so on whatever application you develop will be fragile. It will be fragile, it will be monolithic, it will be monolithic.

And maybe sometimes you may even over-provision, you may even over-provision, some of the systems making it extremely difficult. These overprovision systems - making it difficult to debug. You will have problems of debugging, maintenance and so on. All these issues will come up. This I am sure has motivated you to say that let us look at multi-core systems. Let us look at multi-core systems. But that is still some distance away because now you have actually deviated the topic.

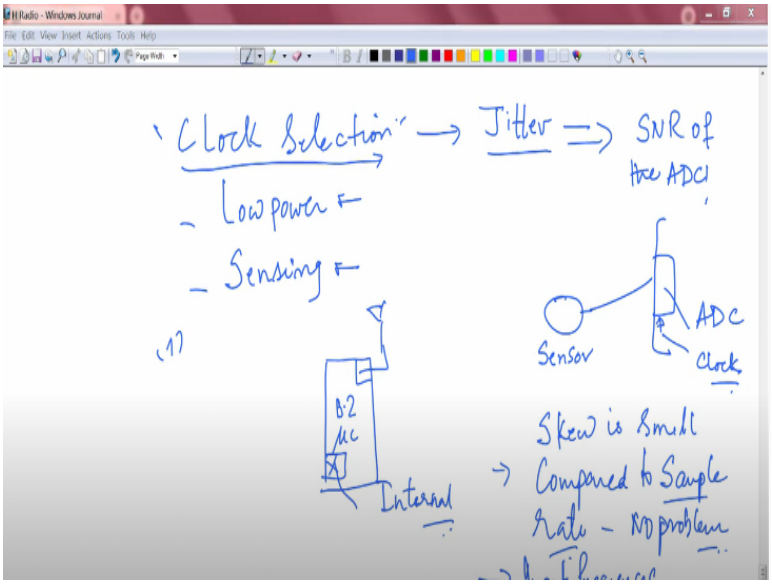
We said let us see how to choose a controller. From the choice of controller we went to motivate ourselves why you need multiple controllers. So, we have still not answered that question of how to choose a controller. So, again I will go back to our slide here. So, now this is done. We now know why you need single and multi processor systems. At least that is clear. Now let us look at the parameters associated with the choice of the controller.

See folks, I did talk to you about the process data sheet which clearly talks about the micro amperes per megahertz. What is the kind of current consumption? Typically it will be in micro amperes per every 1 megahertz of the processor clock. What is the current consumption? Now the clock is an issue and we should go into some detail of the choice of the clock on the controller.

So, again let us exit and go into this direction which is what should be the clock source for the controller? See this is a course on design. So, you should know if I had to choose a controller why do I need to process controllers. What should I choose as my clock source? And what is its implication. Then you should say that is definitely one way. The other way is how does this processor compare; in its performance.

Then what are the typical applications of multi processor and single processor applications? Are we going to put it in industrial applications? Or are we going to do it for control and supervision or is it structural? What is the kind of application we have in mind? That will actually determine the type of controller that you are talking about. So, now let us get into this clock source which can be any of these three different types. If you look at clock selection now, let us do that clock selection.

(Refer Slide Time: 17:33)



The clock selection, again there are two things that we are doing in this course. Very carefully if you observe we are trying to look at low power and we are also looking at sensing. Every time it comes down to sensing. Clearly anything that you do in this course particularly it will have implications on power, it will have implications on sensing. That is why we discussed so much about the ADCs and so on. So, again clock selection is back to sensing as well.

Now the clock usually has to be stable and there should be no jitter. Because this jitter, what does it do? It impacts the signal to noise ratio of the ADC. That is the problem. So, if you have a higher SNR what does it mean? You have more of signal and a very small amount of noise. Essentially you will have no problem. Higher SNR means there is more signal. So, coming to the point clock jitter results in sampling or the sample will actually have a skew in the sample.

You have an ADC, standard picture and you have a sensor connected. If there is jitter this is ADC. Let us write it neatly. I am just showing you the ADC block. This picture has to be made ADC and then one of the channels I am connecting a sensor. It has to be joined. So, if you have jitter in the clock, the clock will be fed to the ADC. This is the clock. Who is generating the clock?

Clock is coming from either an internal clock or an external clock or it is an onboard generated clock. That means you have a crystal oscillator clock which is external to the CPU, to the SoC but it is not external means. External has two connotations. One is the clock. It is not part of the board itself. It is coming from some other source, maybe some other clock generator. Let us say you are talking about two boards.

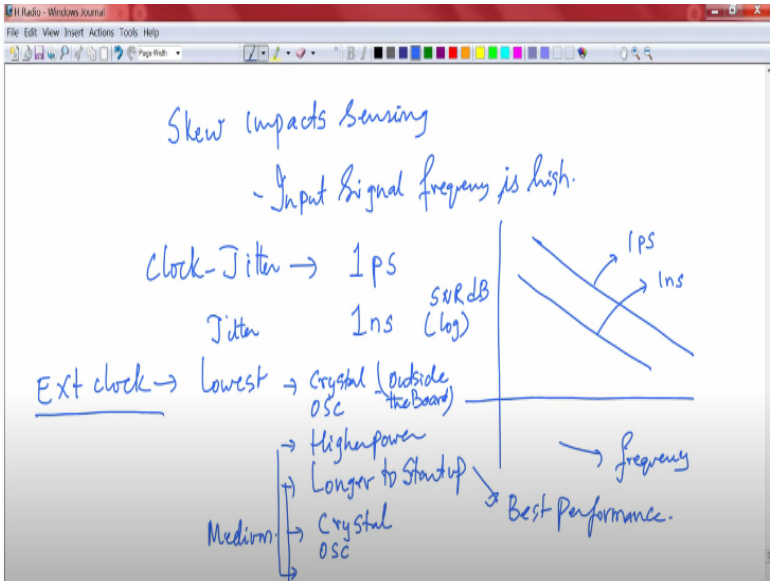
This clock is coming from another board, board one and board two. This is where your controller is. This is where your radio is. This block is purely an oscillator block, which is outside. It is not part of the board. This is one type. Second type is. No this is too much. There is no that kind of space. I will generate the clock using two pins over which I can connect the crystal. This is the second type. Third type is, no this is also too much of components.

Let me use internal. Now here is the problem where you can always use the internal why do you want to use a crystal oscillator outside or you want to use an external oscillator? All of them will have a bearing on the sensing. This is the key point I wanted to tell you about as far as choice of the clock source is concerned. So, let us now go back and look at what are the issues associated with the jitter that comes from this.

See as long as the skew is small compared to sample rate. If it is small compared to the sample rate then it does not matter. No problem. I will simply say no problem. It does not impact much. But higher the input frequency, signal frequency the more jitter reduces the SNR. This you have to be very clear. Why are you choosing a very small skew compared to small sample rate? That is because your input frequency may be very low.

Maybe even close to DC in which case you will not have much of an issue. But if you have an input signal frequency which is quite high then more jitter. If you have more jitter, so I will say input frequency of the signal. Let us write it in the next sheet. I think that is better.

(Refer Slide Time: 23:35)



So, let us say skew is impactful. Skew impacts sensing via the ADC if the input signal frequency is high. If the input signal frequency is high. I simply say it is high. Then it will pull down the SNR. Now you can look up several data sheets to come up with what may be an ideal jitter that should be there. People talk about jitter in picoseconds. Clock jitter is in picoseconds. You will typically talk about one picosecond or one nanosecond and all that. So, be careful about the drop in SNR.

So, you can draw a picture now and you can put it in log scale. You can put SNR in the log. We normally explicit in dB log scale. Then you will have a drop as you increase the frequency. As you keep increasing the frequency, I did not show you in logs. I am just showing you that the whole thing is in a log scale. Obviously this must be the 1 picosecond and this must be the 1 nanosecond. So, you can see that performance can matter a lot. What does it all mean to us?

It simply means if you choose an external clock, what is the jitter? It will be the lowest, I will say external. I will simply say jitter. It will be the lowest. It will require a crystal that means more components and any crystal oscillator is associated with higher power. So, you see everything is boiling down to sensing and power. So, this will come down to higher power and one more nuisance with the crystal. It takes longer to start up.

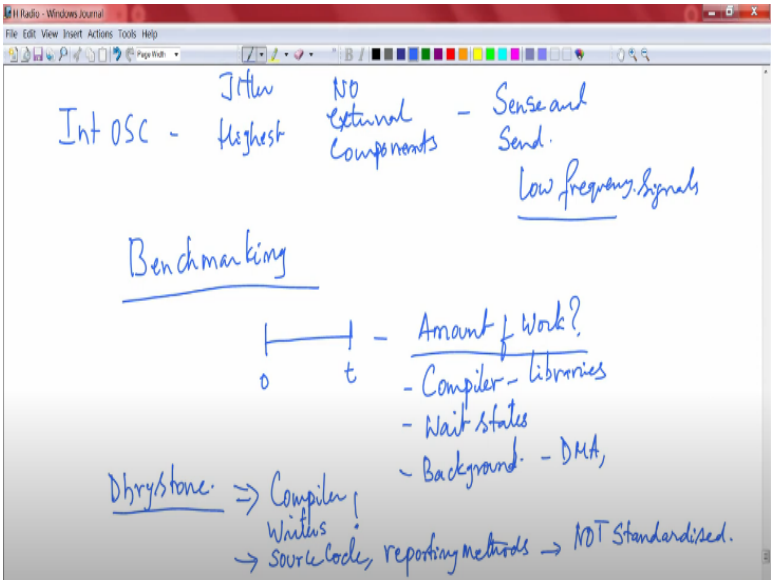
Supposing you put the controller to sleep you are using an external crystal and you put the controller to sleep. What happened? The crystal oscillator has to come up, that oscillator block has to come up and then it has to start giving you the clock signal. Now that takes time and during that period you cannot do anything number 1, number 2, power consumption also will go up. These problems will be there.

So, you can choose the external clock carefully whether you want to do it that way or not. But I can tell you for sure this gives you the best performance, fantastic right?. You have the best solution on hand. Then you can also do medium jitter. In the medium jitter case this also requires a crystal. But this crystal is where? It is just placed outside the SoC. In this case you do not know where it is. It is coming from outside the board.

So, please note that the external clock means it is coming from outside the board. So, I will say outside the board. You are generating a fantastic very stable clock and then you are bringing it. I mentioned about board one and board two if you recall, maybe coming from board one. Here in the medium you are still doing single board and the same problem will come. You can put ditto here and you can put ditto here. Longer startup and all that.

But then if the application is still the same the jitter is medium. That means again you have to take care of safety margins whether it will impact the sensing or not. The quality of the signal that is sensed is impactful or not will have to largely depend on what you want to do. Now the third one is the lowest medium and the other one is highest.

(Refer Slide Time: 29:44)



Internal oscillator if you choose this will give you the jitter which will give you the highest jitter but advantages, no external components. And it is good if you are talking about simple sense-and-send. So, again it is connecting back to send. You have simple low frequency signals almost DC then you do not have a problem of low frequency signals. Well, you can go on like this and talk about controller and implication of ADC, implication on the reference for the ADC and so on.

But they are all a little bit out of place here. We will not talk about that because I wanted to concentrate back to get you back on track here. So, folks, now you see, we exited here. We have to come back here again. And when we come back, we realize that the story does not end here. You have to go back and look at benchmarking also. That is the third thing that you may have to worry about. See folks, benchmarking is another story.

Now when you talk about benchmarking how will you choose the controller from a performance perspective? That is what you want to do. Suppose you have a certain time period. This time period starts from zero and stops at t. This is your time period. Now you are asking the question what is the amount of work done in this period. That is not a trivial question. You cannot answer that question. I will tell you why. Suppose you have a compiler.

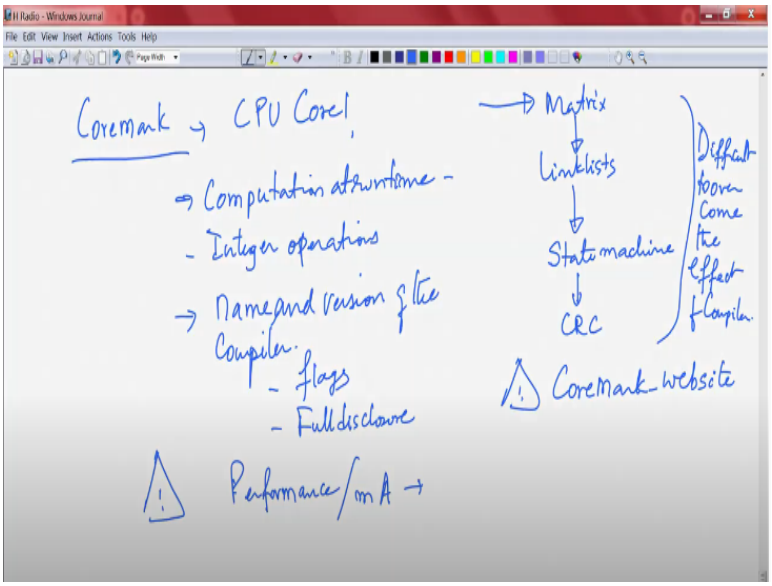
Now compiler is actually dependent on many factors including compiler. It will be dependent on how the compiler was designed, it will depend on the different libraries, it will depend on the wait states if they were introduced and what about background activity? What does background activity mean? What about DMA? Maybe DMA is happening in the background. And so many things have to be looked up when you talk about this question, if you want to answer this question of what is the amount of work done in this time period.

So, it is almost difficult for you to arrive at something useful. Now it does not end there. People have proposed two ways of sort of benchmarking CPUs. One way is to use what is known as Dhrystone. But unfortunately, this Dhrystone is I think interferes quite a bit into the compilers. It interferes quite a bit into the compilers. It is more useful for compiler writers. Not so much for compiler writers I would say.

So, this arrow is to be replaced by saying you are good, but it is for compiler people not to benchmark a CPU. Because time it takes for executing a library function is actually not accounted in Dhrystone. Because it is going out somewhere and picking up something and then the whole thing is not easy. That is one part. Second part is they do not give you any simple way of the source code.

And the reporting methods, both have a problem because they are not standardized. So, this will create a confusion in the minds of people when you do not use them in a uniform way across processors.

(Refer Slide Time: 35:27)



So, the newer standard that has come up is the Coremark. Good thing about Coremark is just talking about CPU Core. He is just talking about the CPU Core. He is just talking about the CPU Core. And this is very important. All the computations are made at runtime, all computes. So, here you see. So, no question of going to a library function and all that you do all computations at runtime which is not the case in Dhrystone. If you put this condition the compiler is actually stuck.

You cannot get in the compiler, you cannot do any escape out for the compiler because it cannot cleverly solve the problem if you are doing this at runtime. So, is it mainly focused on, this Coremark is mainly focused on integer operations. One other thing is we mentioned about the use of standards in a standard way which was not the case in Dhrystone. Both the source code that is used and the reporting methods that was a problem, you have to somehow include the name.

Each benchmark should include the name and version. It must include the name and version of the compiler. You have to disclose for sure. You must put down the compiler flags, you must mention that. That way you will catch. Full disclosure should be possible, full disclosure, no hiding and so on. And therefore, you are more or less you cannot go around taking advantage of compiler and compiler library calls and so on.

You cannot do that because everything is done as I mentioned it is in runtime. So, the way Coremark works is it takes very standard operations. It considers a set of standard operations. It could be matrix manipulation leading to some link lists leading to a state machine and then maybe CRC computation. When you do things like this it is almost impossible to overcome the effect of the compiler. So, no effect or difficult to overcome the effect of the compiler.

So, that is a good thing about the Coremark. So, folks, I want you to go and look up the website, Coremark website. It is rich in information and choose it. Use your benchmarking whatever you wish to benchmark, do look up those codes, run them and decide on the performance of the CPU. This is very important. So, of course, the Coremark is highly dependent on clock rates. There is no doubt about that. But that is where you are trying to differentiate between one type of processor and the other.

Is not it? So, you must look at this Coremark website to give an understanding. But I must also tell you that things are not just as simple as what you think. There are other things also you have to consider. One of the things that people are looking up in terms of Coremark is performance per milliampere. Everything revolves around the current consumption. This is becoming very important.

Microwatt consumptions are playing a very important role in deciding on how many microwatts per megahertz does it takes, for instance is something that people are worried about.