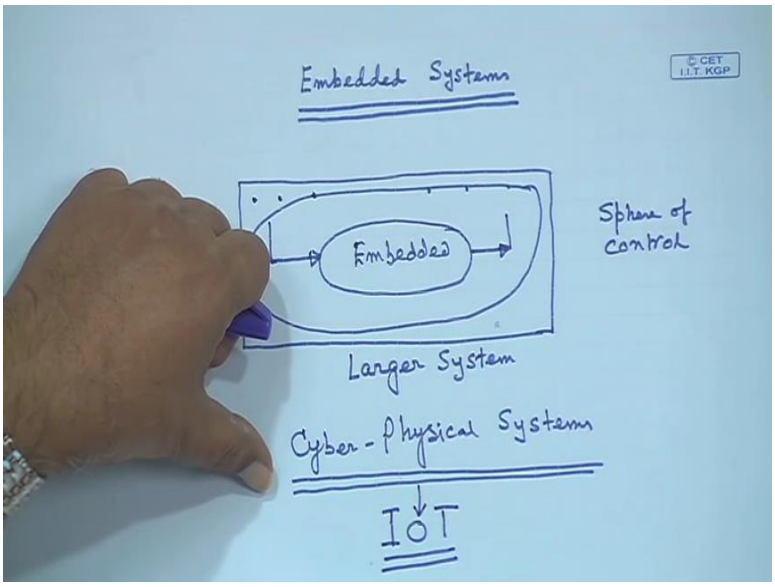
Embedded Systems Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 01
Introduction

Good morning. And welcome to you all to this course on Embedded Systems.

(Refer Slide Time: 00:27)



We will try to understand in today's lecture which is the first lecture, how embedded systems differ or how embedded system poses some additional constraints over the other general purpose systems that we have encountered in our learning process till now. Now embedded system, what is an embedded system if we ask this question what is an embedded system? The possible answer is an embedded system is a system where a microcontroller based or microprocessor based programmable system is embedded in a larger system. So, you will have a larger system we can be; this is the larger system in which there is a microcontroller based or programmable system embedded always it may not be programmable by the way, it may be also hardware programmed.

So there is some processor embedded inside this, which interacts with this larger system at different level. So, this system actually takes inputs from the world and by the world I mean this environment with the world for this embedded system. So, it takes input from this world processes the information, sends it back to the world. It can come from different points of this larger system and the after computation the responses can go to again different points of the system.

The points which are affected because you see the larger system can have many things everything may not be controlled by the embedded system everything cannot be sensed by the embedded system. So, the parts which are sensed acted upon and actuated upon are known as often known as a sphere of control. So, might be there is a sphere of control around this embedded this embedded system, within this larger system that is being affected by this embedded system.

Now, typical examples of embedded systems now you can think of embedded systems are everywhere; everywhere you can say if you take your mobile phone it has got many communication systems, but also there are processors even earlier if you take the printer: inkjet printers, colour printers there also the controllers are there even. If I go even earlier, there are some discs systems where intelligent controls are embedded printers I have already said.

So, there are there have systems been embedded inside larger systems for longer time, nowadays we find intelligent washing machines where depending on the volume of the clothes, the extent of dirt in the clothes, the gyration, the speed at which the there will be centrifuged or how much detergents will be required all are controlled automatically; how is it controlled automatically? Using some embedded systems.

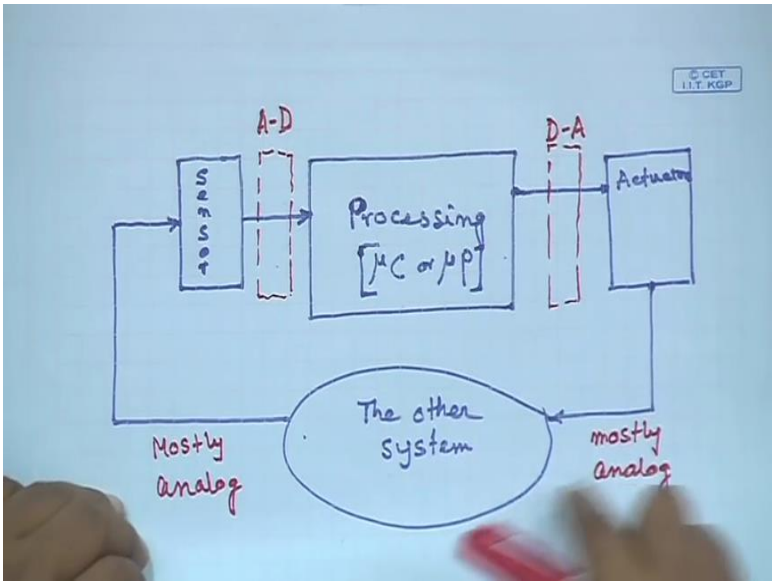
Again if we go to more advanced things, now most of the modern cars are coming with not one embedded system, but multiple embedded systems, which some are taking care of controlling the brakes some are taking care of the cruise control, some are taking care of the locking system. So, there are number of embedded systems impregnated inside the largest system that is the car. If we look at the airplanes, most of the airplanes now are said to be not so much dependent on the pilot's. So, there is a set of embedded controllers

inside that which can negotiate with the environment and those are also embedded systems. In warfare where we find the when we want to have some rocket launchers, there also embedded systems are there which are taking signals from here. So, the list gradually becomes endless nowadays.

Now, if you look at, now this is also known as sometimes a variant of such embedded systems are known as cyber physical systems, this term is becoming popular day by day, where the cyber part means the computer in interacting with a physical environment. So, the physical environment can be a road on which a car is moving right the road will have different frictions at different a point that is the physical reality and now I have got a car, which has got a controller which will keep the speed fixed at a particular set level.

Now, from your school level knowledge of physics, you know that in order to keep the speed constant when the friction increases you have to apply more force and the reverse when the friction goes down in order to keep the same speed. Now that the physical reality is the road, the physical reality is the condition of the tires of the car and the controller is within the car this is one sort of thing one sort example of cyber physical system this can be farther extended nowadays to the term which many of you have heard which is internet of things I o T as it is told. I o T is nothing but a set of systems spread around maybe distant, each of them have got their embedded controllers, embedded sensors everything and they are communicating over the internet and one system can communicate and actuate another one at a distance. Therefore, everything gets connected through the communication of a set of embedded systems.

(Refer Slide Time: 08:34)



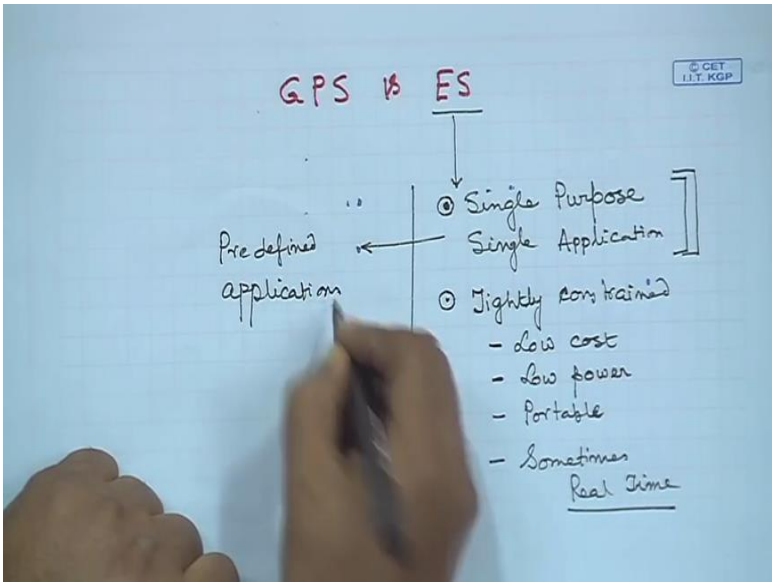
Now, coming back to embedded systems proper, if we think of an embedded system it will consist of say a microcontroller based system or processing system. Let me call it the processing system, which usually are microcontroller or microprocessor based if they are programmable and on another side we have got the sensors, which interacts with the world and provides the data to the processing unit and there are actuators. Now these actuators are actually affecting the physical world or the largest system, let me call it the other system whatever that the other system might be alright and the sensor is sensing some variables from this other system and feeding it to the processing unit and depending on whatever actions are to be taken there the actuator is actuating it.

Now, in between I have missed out certain things intentionally that is it can the sensors are sensing the data, which are coming from the other system which is a physical world, which is mostly analogue mostly analogue right? If it an analog systems. Therefore, the there can be 2 things; the sensor can directly sense the data in the digital form, otherwise if it does not do that then here we need a layer of interface which is the analogue to digital converter, because this processing system is a digital machine. Similarly we have to give the data to the physical environment mostly analogue may or may not be, but if it be an analogue system then we need here a D to A converter, digital to analogue

converter. Now often the actuators can be direct to digital and can provide the control that will see some examples later.

So, this is the overall environment. In order to learn embedded systems, we have to learn some; we have to have some knowledge about the sensors, what sensors, how do you specify sensors may be many of you know how analogue to digital converters or digital to analogue converters are working we will try to brush up on that again also again. And most important part is this processing unit, where we will see that different types of processing's can be involved; now you one can ask the question that is in a typical architecture of a computer system, we also have got the input device, the CPU and the memory, the processing device processing block and the output device. So, where does it differ from a general purpose system?

(Refer Slide Time: 13:07)



So, how does it differ from the general purpose system? So, general purpose system versus embedded systems; usually the embedded systems are one embedded system is for one purpose. So, single purpose or single application that is point number 1, the other point is the embedded systems is severely usually what happens here, what do you mean by this? A single application program is repeatedly run, whenever for a particular say for example, let me give the example of an answering machine, telephone answering

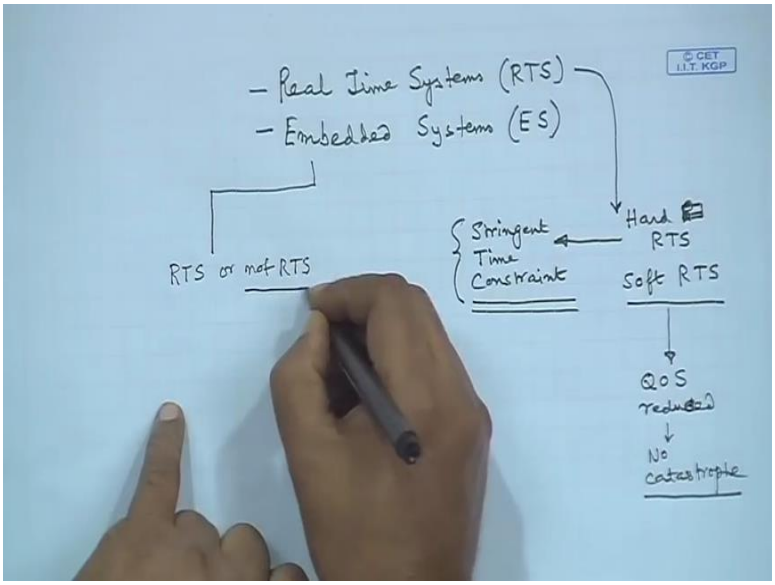
machine, it has got a fixed task that when the call will come after 5 or 7 rings, if the receiver is not picked up see all these things are to be sensed that the receiver has not been picked up, then the answering machine goes on after the delay gives up, after plays a message and gives a beep, after recording for a particular delay it cuts it off all those things.

Now, this is the fixed this program, it has been written once and have been put in an embedded processor, which has been put in the answering machine or the telephone system. So, that program is repeatedly working.

The other option is other distinguishing feature is; it is very tightly constrained compared to the general purposes system. General purpose system is not single part single purpose; it will be multiple purposes it can be used for different purpose now is tightly constrained in terms of what? It must be low cost, the cost cannot be exorbitant like a general purpose server right, it should be low power mostly we will like to run this on battery it should be portable, it should be sometimes it must be real time. In general it should be fast, it should be fast all the time, but sometimes it should be hard real time I will come to that what it means. So, these are some of the constraints that are specific to embedded systems, but not so much for general purposes.

On the other hand general purpose systems are more powerful in the sense, that it can do many more things it is not single purpose, is not constrained by these things, it consumes more power usually runs from the direct electric supply with some battery backups and all those. So, that is the differentiation between general purpose systems and embedded systems, most important thing is its predefined what I am going to run on an embedded system is predefined application. In some cases we will find that multiple applications are running on an embedded system, but those are also predefined 3 or 4 it cannot be just like a general purpose system that you can go on adding to this.

(Refer Slide Time: 18:00)



I differentiate between another thing that since right now I have said real time systems, let me also differentiate between RTS and embedded systems, what are real time system real time systems? Real time systems are systems it can be of 2 types: hard real time systems and soft real systems; hard RTS and soft RTS. Now hard real time systems have got stringent time constraints, the time constraint that is given is very stringent. Now this stringent time constraint must be adhered to, must be respected. If this time constraint is violated then violations will lead to catastrophes.

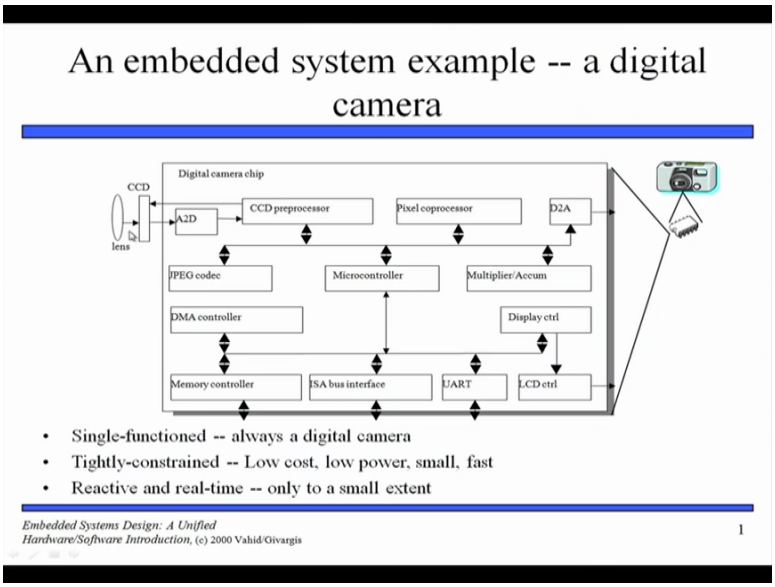
For example you have to land within a plane has to land within maybe 1 minute alright? Now if it exceeds 1 minute then it can crash; take another example say a nuclear power plant or some power plant some thermal plant, where you find that the temperature has gone very high beyond the safety limit and you have to start the coolant and put the power put the shut off the plant within some specific time and suppose your system that is controlling it is sluggish enough that it does not respect that time it does the thing, but does it much later by that time the situation has blown off right.

So, that is not allowed in hard real time systems, why we call it real time systems? Because this time constraints are specified in terms of real time, real time means what? 1 second 2 second by this clock, this is the real time clock time is ticking. On the other end

what is the other sort of time that we discuss about? We discuss about so many cycles, suppose I say that the light must be switched off after 50 cycles, visa vis I say that the light must be switched off within 1 second or 1 minute or 1 second what is the difference? If the cycle clock is 50 hertz then they are equivalent, because I have said after 50. So, it will be fine, but if the clock is slow then 50 cycles will come after 2 minutes right. So, that is the basic difference between real time systems and normal I mean that typical computer systems that we talk of.

On the other hand hard real time soft real time systems, if we violate the time constraint then we the quality of service gets reduced quality of service reduced, but no catastrophe happens no catastrophe happens. Now, embedded systems can be real time systems or may not be real time systems. It may be a real time system or may not be a real time system again on the other hand similarly real time systems maybe embedded usually real time systems are embedded, but real time systems can also be same server based system. So, these 2 are not equivalent that all embedded system should be real time systems that should be kept in mind.

(Refer Slide Time: 23:14)



Next we will come to a slide that will give you an example of an embedded system here. See this is an example of an embedded system which is a digital camera it has been taken

from the book embedded system design, a unified hardware software introduction by Frank Vahid and Gavages it is Wiley publication.

So, here you see typical digital camera, what do we have inside. Now first of all it is a single functioned its single functioned its only one function, that is been done that is photograph is taken and stored right. It is tightly constrained how it is low cost, otherwise people will not buy it, low power it is small portable must be portable and should be fast.

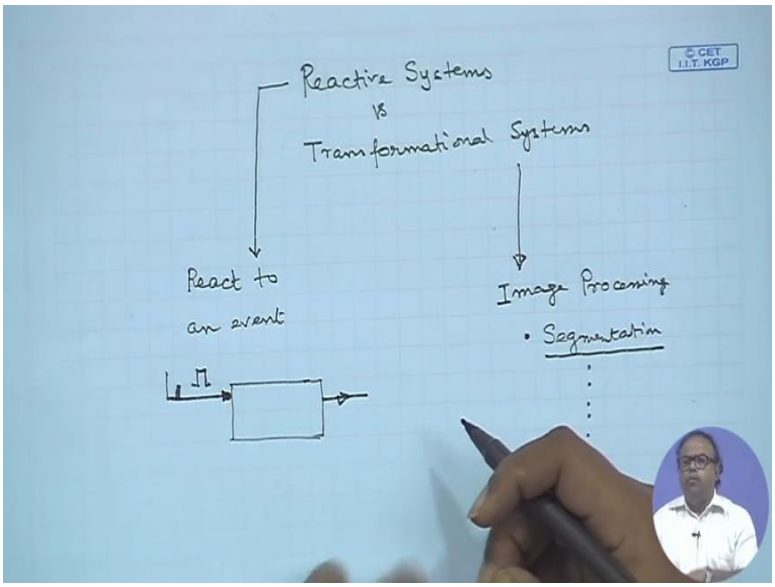
Now we do not call it real time systems, if the picture is captured a little slow no catastrophe will happen right it is reactive and real time I will explain what is a reactive system and real time system, but it is a reactive system and real time only to a small extent, because if you take the image it is the soft real time or hard real time? It is the soft real time; obviously, because if it is very slow then probably you will not buy this, but no catastrophe will happen.

Now, what is there? You can see the lens which is capturing the world through a CCD charged couple device, there is an A 2 D converter which is converting it there is a charged couple device pre processor then here you see one thing that I in the slide that I was showing about the embedded system architecture also you can see that there is a communication channel going on also this, that I did not explicitly mention, but communication is another very important part of this entire embedded system ecosystem. So, here you see that through the bus we are communicating the CCD processor data is going everywhere and there is a microcontroller there is a JPEG codec, which is doing the conversion and completion and also there is a DMA controller, memory controller, bus interface, UART is a serial parallel to serial converter and there is an LCD control.

So, there are so many components inside an embedded system besides a processor. Now you see how many processor do you see here? You can see a microcontroller is a processor here, you can see a CCD pre processor, Pixel core processor, here is another JPEG processor, multiplier you can say that this is not a processor it is just a multiplier, but that is also doing some processor. So, there is many processors communicating now

the key point is that some of these can be hardware can be implemented in hardware, some of them can be implemented in software will see this part.

(Refer Slide Time: 26:52)



Next what are the design challenges, before I come to the design challenges let me make another system level classification, one is reactive systems versus transformational systems. Reactive systems are those which react to an event that occurs for example, there is a system like a camera it does not off itself work whenever you create an event you there is an input and in that input you put a pulse like suppose you are pressing the shutter, then only it starts work.

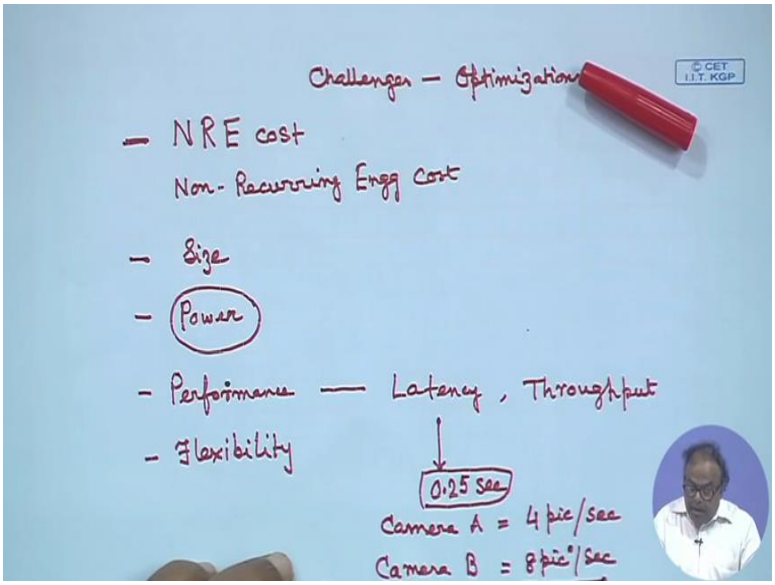
Similarly, it could be that this is continuously it is keeping watch its active another system, but whenever there is an alarm situation then only it starts working and immediately it sends the response and usually such reactive systems in many cases such reactive systems react to an event, which is an urgent event, and therefore in many cases they are real time systems.

On the other hand; so these are working on external events, on the other side there can be transformational system for example, image processing systems. You get a set of images, you get an image and you process it in image processing you do a lot of things, you do

noise removal, you do segmentation right and you do go on doing all these things different activities that you do right? Now that you are taking a data some data is coming in, and that data is being transformed through stages, not in the event of any not in the event of some explicit event asking for service. Here if we go to this diagram, you can find that there will be some reactive components also here and some will be transformational component so often the system is a mix of this.

Now, embedded systems have got a number of design challenges, a major design challenges the cost obviously.

(Refer Slide Time: 30:10)

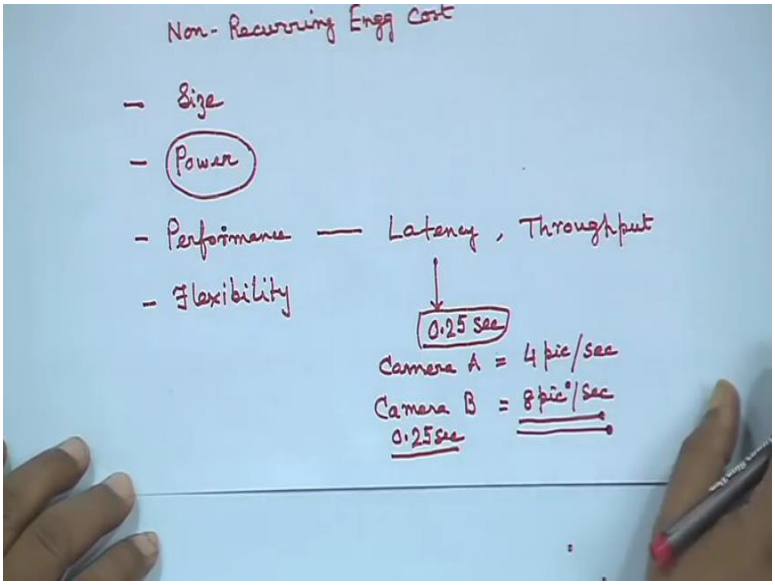


Cost is a very important aspect now; the cost has got 2 components: one is the NRE cost, NRE cost means non recurring engineering cost; we will have to incur some cost for that. So, here are the challenges, the challenges are mostly the challenges and optimizing on different aspects. What is the cost I mean how much time how much how much money I spend in order to do this development and that is also dependent on the time, this is also dependent on how much time I will give size of the device the system, the power, power will come back will come back to power time and again power is a very very important factor in embedded system design nowadays. The performance and flexibility what do I mean by performance? Performance means how in regard to time, how fast does it work?

For example and it can be latency, it can be throughput, what is the difference between these 2? Latency means the time to take a task let me give an example suppose the camera that we just now saw, takes 0.25 seconds to take a picture. Now I have got a camera some camera A. Therefore, can take 4 pictures per second right because I have designed it in such a way that it can take a picture within 0.25 seconds, process it do everything.

Now so, its throughput is also 4 pictures per second; latency is 0.25 this is latency, and this is the throughput. Now there can be another camera B which has got the same latency, but it can take 8 pictures per second, how is it possible?

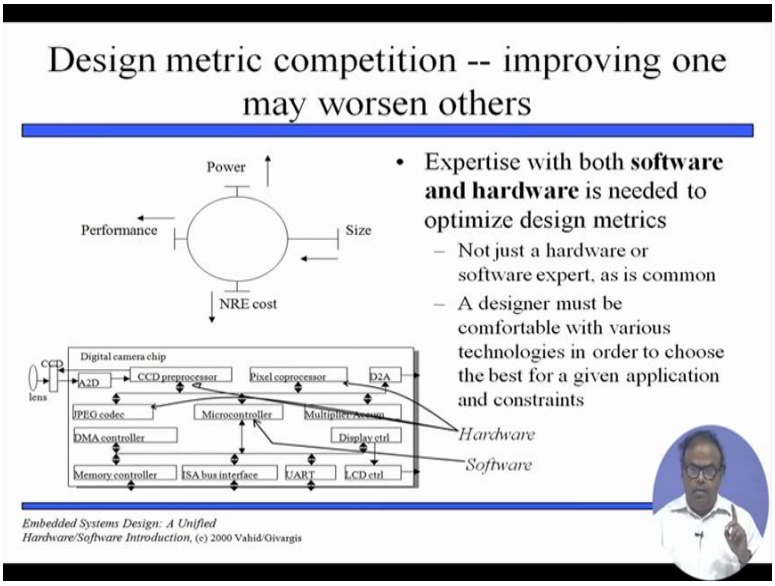
(Refer Slide Time: 33:35)



So, this is the throughput and this is the latency, I said the camera B has got the latency of; camera B also has the latency of 0.25 seconds right. So, the camera B takes a picture, in order to take a picture it also requires 0.25 second just as camera A did, but camera B had 2 threads running to parallel processing. Therefore, while one is being processed it could be pipelined, one is being processed another is being fetched. In that way I can enhance the throughput, but the latency is same, I hope that differentiation between throughput and latency is clear. So, that is performance.

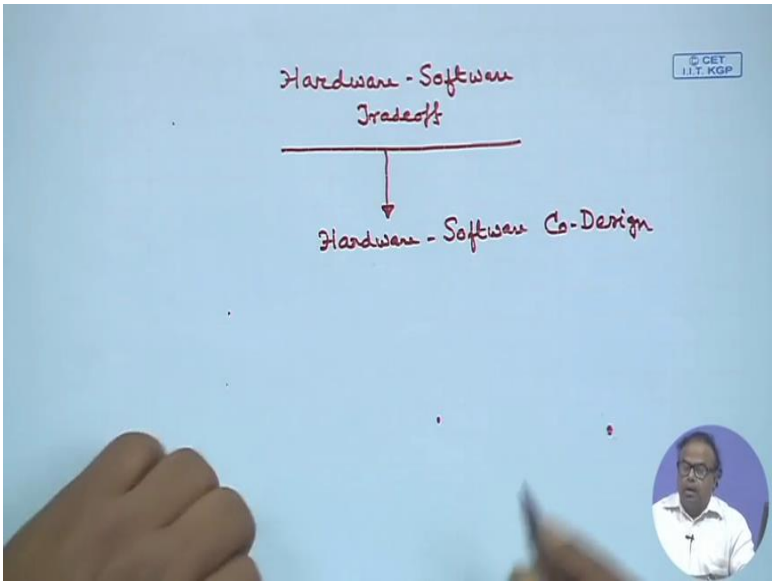
The flexibility is a very another important thing and flex if I bring a product, which has got the flexibility of adapting to. Suppose now I have got I bring a phone, which can work in 4 g as well as 5 g, visa vis there is a phone, which is coming in 4 g or only in 5 g. Now there is a feature differentiation, my phone which I am bringing now can work in 4 g and 5 g verses your phone which works only in 5 g. So, I have got a flexibility, by that I can greater handover the market that is one. Another point is that if I have got the flexibility, I bring the product in the market and then I find that there are some more demands coming out from the user, can I adapt to those demands? If my system is flexible, I can add new features to the system and make it more flexible. So, these are some of the challenges that we have to face.

(Refer Slide Time: 35:52)



Now in order to complete the discussion today, we will go back to another very important issue. The design matrix are all these the energy cost and all those things, energy cost you should try to bring down, power is another now each of these constraints are competing among themselves, they are competing among themselves. Therefore we have to always find an optimization right there is no unique solution.

(Refer Slide Time: 36:32)



Now, let us look at this diagram, what I am talking of now which is a very vital factor in embedded system design is hardware, software, trade off and that is why embedded system design also goes by the name, the task also goes by the name, hardware, software co design both of these things are designed hand in hand. Now let us look at this picture here in the digital camera chip we have got so many functions to achieve, now this CCD processor pre processor, I can map it to a hardware the JPEG codec I can map it to a hardware, or this JPEG codec had the option that I run a software code on the microcontroller for having the compression.

So, I have got an option of doing this compression in hardware or in software, similarly the pixel coprocessor could be done in a software or could be done in a hardware, what is the advantage if I do it in a hardware? It will be faster, what is the disadvantage of doing it in the hardware? The cost the area it will become more, it will become heavy all those factors will come into play. What is the advantage of making it in software? It will be flexible, I can tune it further, but what is the disadvantage of doing it in software? It will be a little slower and there will be the of course, I will need a microcontroller and power consumption; power consumption will come will deal with that separately.

Similarly, this, but some of the things that for example, there a bass interface that is hardware. So, some are fixed assigned destined to be hardware, some are destined to be software, but some can be implemented by either hardware or software and the designers challenge is to really decide on which one should be done in hardware and which one should be done in software. So, we will take it up from this point, we conclude today.