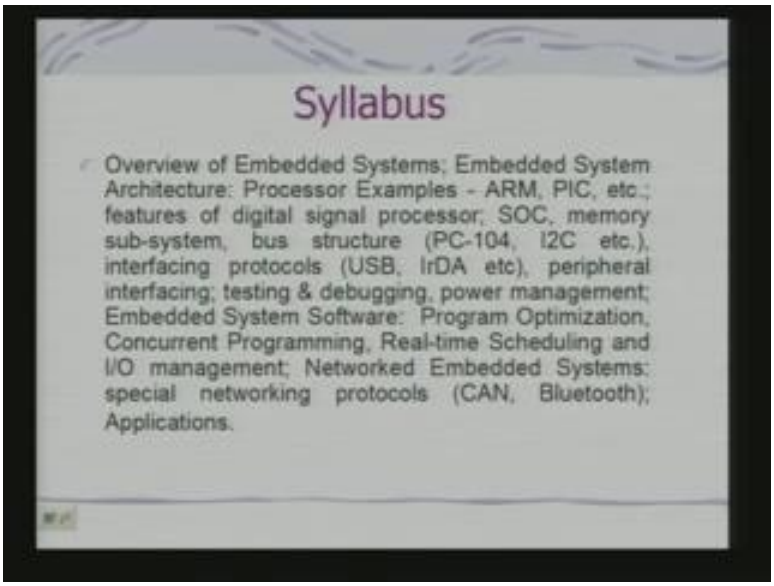
Embedded Systems
Dr.Santanu Chaudhury
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Lecture - 1
Embedded Systems: Introduction

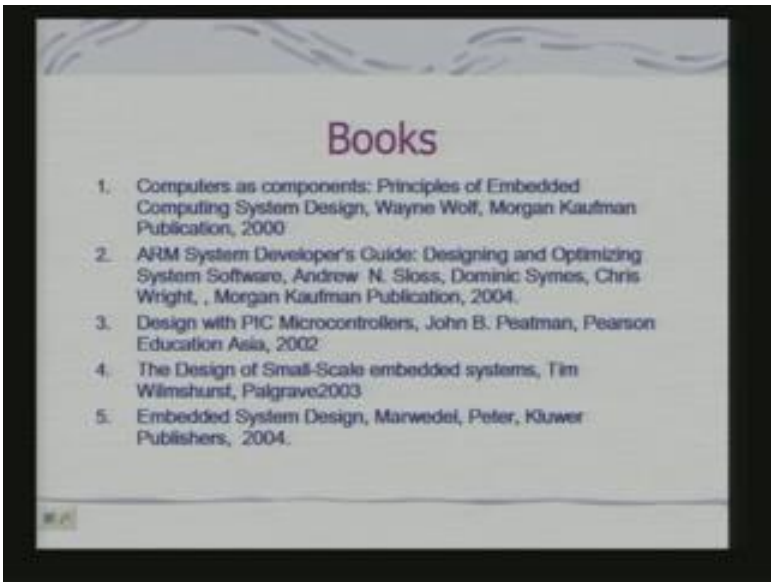
Welcome to this course on embedded systems. Today, we shall discuss what embedded systems are and before going into what they are, let us look at the syllabus that we shall follow.

(Refer Slide Time 01:35 min)



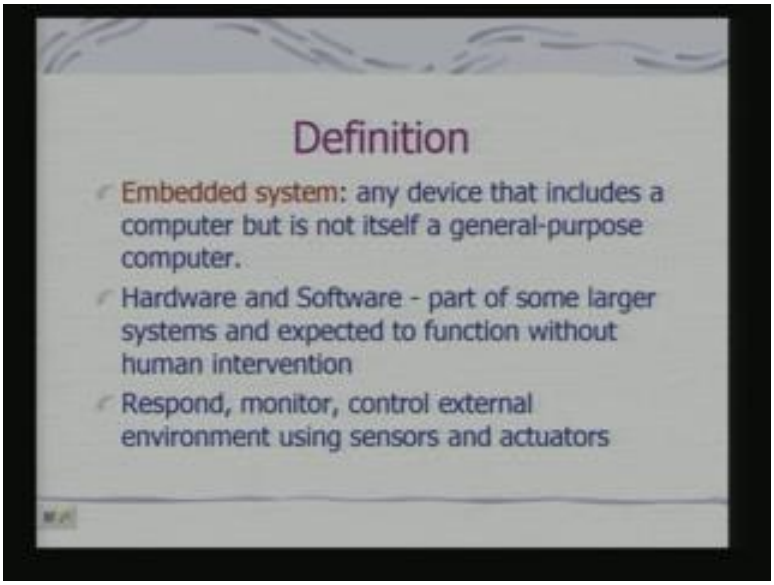
This provides a broad outline of the course that we shall be covering in 40 lectures. We shall cover the processors, bus structures, interfacing issues. We shall look at software, optimization of the program, also real time OS issues. We shall have occasion to examine different aspects of network embedded systems as well. These are some sets of books that you may follow. There are various other books available in the market; you can refer to them as well. But these are the five books which I shall primarily follow in this course.

(Refer Slide Time 02:11 min)



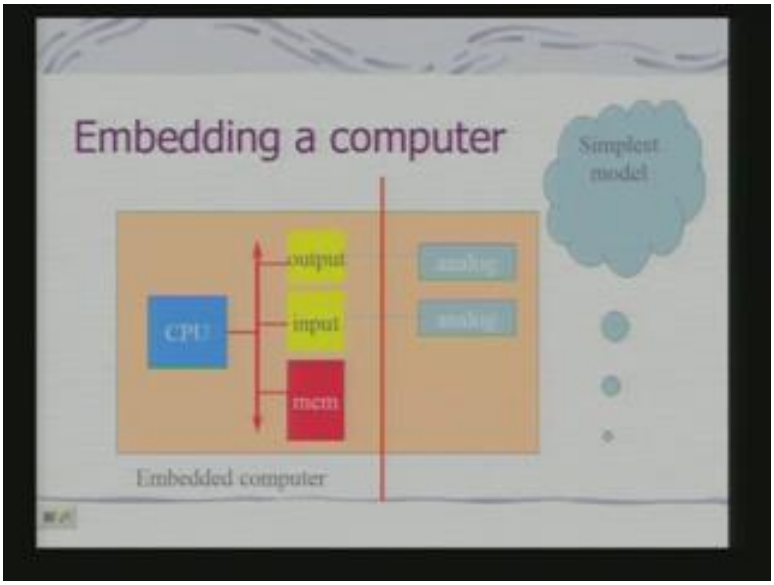
Let us start with the definition of an embedded system. What is an embedded system? Any device that includes a computer, but is not itself a general- purpose computer. It has hardware as well as software and it is a part of a larger system and is expected to function without human intervention. An embedded system is expected to, expected to respond, monitor as well as control external environment using sensors and actuators.

(Refer Slide Time 02:34 min)



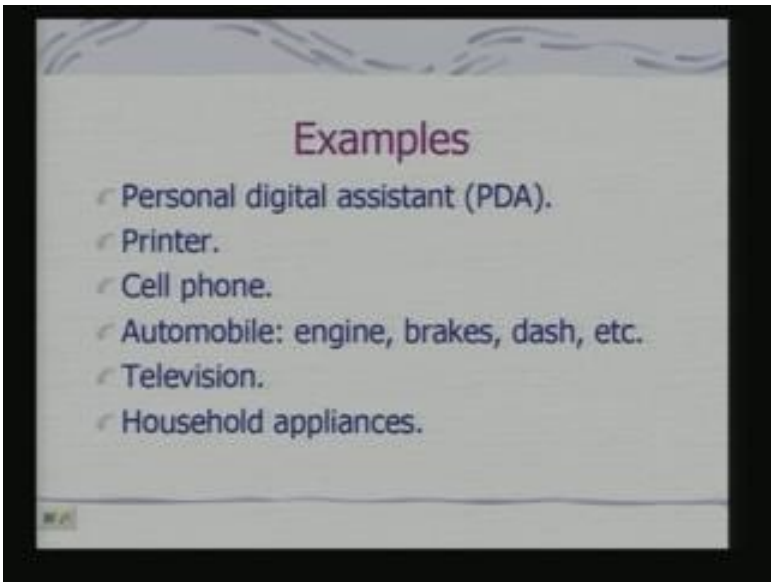
So, basically what we are talking about is embedding a computer; embedding a computer into an appliance and, that computer is not expected to be used for general purpose computing. Since it is embedded into an appliance, it needs to interact with the external world, so it has got analog interfaces. And the model that I am showing is the simplest possible model that you can have of an embedded system.

(Refer Slide Time 03:09 min)



Let us look at the examples. Examples are, personal digital assistance, printers that you use in computers, cell phone that all of us are familiar with. Automobiles, in fact automobiles have got a number of microcontrollers and it is actually an embedded networked computing system. Television; in television for various purposes micro controllers are used, as well household appliances.

(Refer Slide Time 04:22 min)

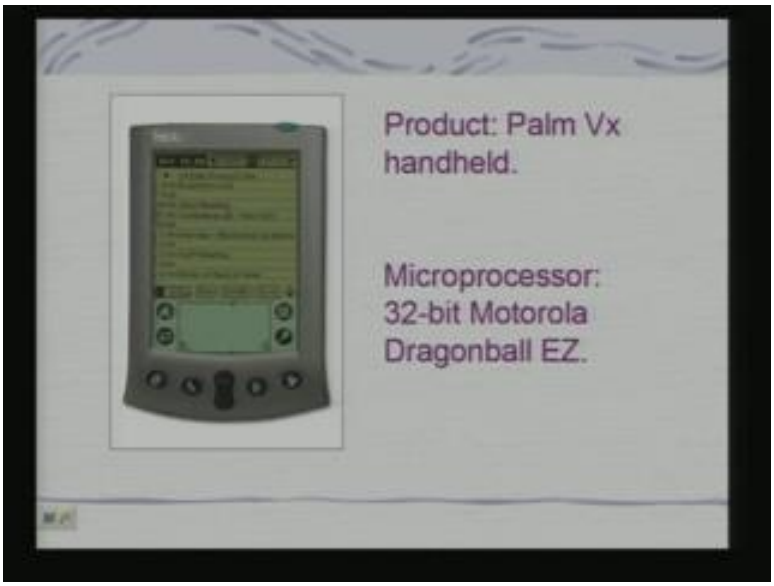


Let us physically see some of these examples. Here we have got a PDA; this is a digital camera, this is a cell phone and all of these embedded systems actually have not one micro controllers sitting inside, but more than one micro controller sitting inside. And that is why managing these micro controllers, designing their hardware, designing the software for managing these appliances, those are different challenge than that of designing a general purpose computer.

(Refer Slide Time 04:37 min)

So, let us go back to the other example which is that of a surveillance system, in fact, surveillance system off late has got tremendous importance because of various security reasons. So your video cameras; your video cameras as well as your biometric systems which are using smart cards etc, they are also part of embedded systems.

(Refer Slide Time 05:44 min)



So, these are again another example of a palm, that is a PDA, and what I would like to show is, in this cases the different types of microcontroller that have been used. This PDA uses 32-bit microcontrollers. Another example is that of a cell phone, it uses also of 32-bit microcontroller. Then, if you look at household appliance example, front panel of microwave oven that also uses a microcontroller, but typically it will have its word size much smaller than that of the earlier examples; because the functionality that it handles is much less. Then, if you come back again to the camera, in fact the camera we had seen few minutes back; that uses a again a 32-bit processor because it handles complex functions. Similarly, in an analog to be a simpler microcontroller, than that in a digital TV, because in an analog TV, the microcontroller handles primarily the problem of tuning and channel selection. But in a digital TV, decompression, disk family and particularly on the set top box, your microcontroller handles a number of complex functions.

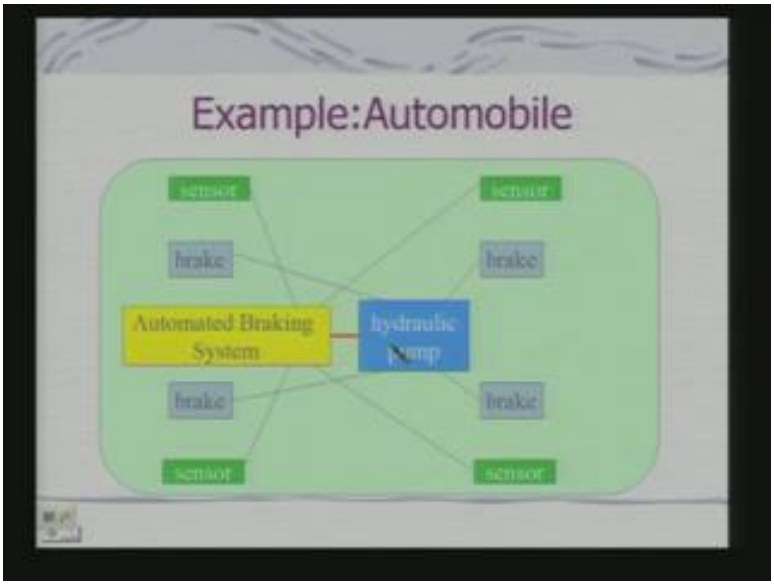
(Refer Slide Time 07:17 min)

Let us look at an automobile system, today sophisticated automobile may have more than 100 microprocessors, a 4- bit microcontroller can check the tension of the seat belt. Microcontrollers can run the display services on the dashboard, also it will control the

engine and since the engine controlling is the most complex function, it has got the most powerful microcontroller that is 16 or 32- bit microcontroller.

Let us look an architecture of such a system, a breaking system; this is another aspect of an automobile. So what we have found here is; so we have got sensors, this sensors actually sensors the speed. And these are the breaks which are controlled by hydraulic pump and your embedded system is this automated breaking system which receives input from the sensors and then depending on the software that is running in the automated breaking system, actuates the hydraulic pump to control the break. So this is an example of a control system being implemented through the help of microcontrollers in an automobile.

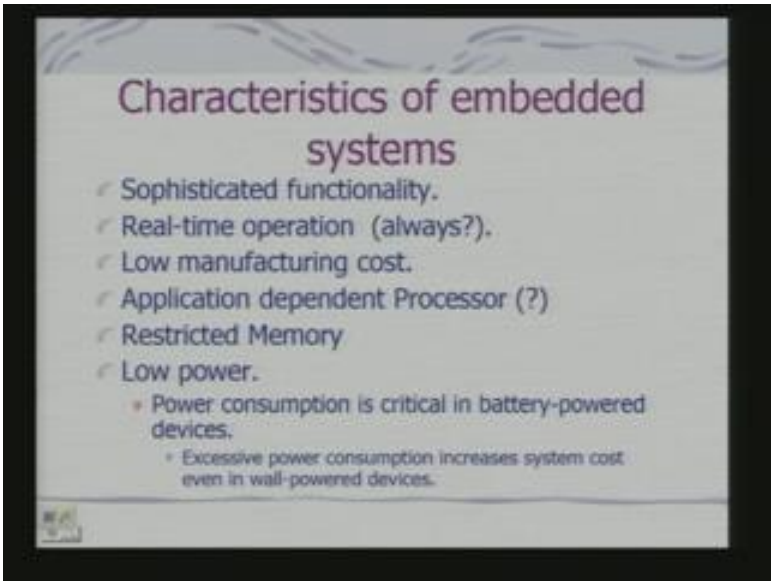
(Refer Slide Time 07:42 min)



Therefore what are the characteristics of embedded systems? First thing is, they all implement sophisticated functionality. The degree of sophistication can vary from

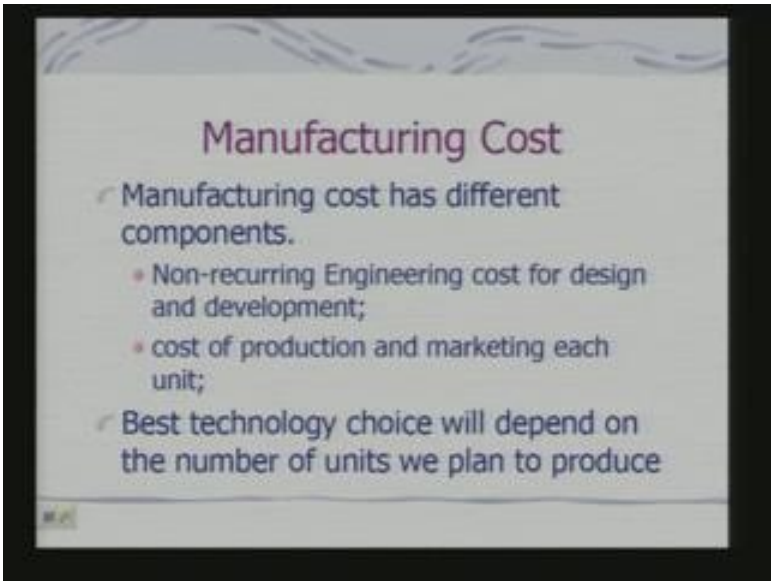
appliance to appliance. They satisfy real time operation. Is it always true? It is not necessarily true and what is the real time operation, we shall comeback to this point slightly later on. They should have in many cases, low manufacturing cost, but cost itself is an issue which requires further closer examination. In many cases these appliances uses application dependent processors and not general purpose processors which we find in computers. They need to work with restricted memory and the most important consideration, is that of a power because many of these devices are actually battery operated. Also when we do not have battery operated devices, that is well mounted devices powered from direct power supply, then power consumption becomes an important issue, because I need to do a heat management, heat dissipation design for this devices which can add on to the cost of the embedded system.

(Refer Slide Time 09:28 min)



So let us look at this issue of manufacturing cost. There are two aspects; first aspect is what we call non- recurring engineering cost, which is actually the development cost into that system. The other aspect of the cost is production and marketing each unit. If we are targeting a mass market then what we need to optimize is a production marketing cost.
But if you are trying to develop a very specialized application then I can invest in NRE as well as I may compensate set for high production cost.

(Refer Slide Time 10:01 min)



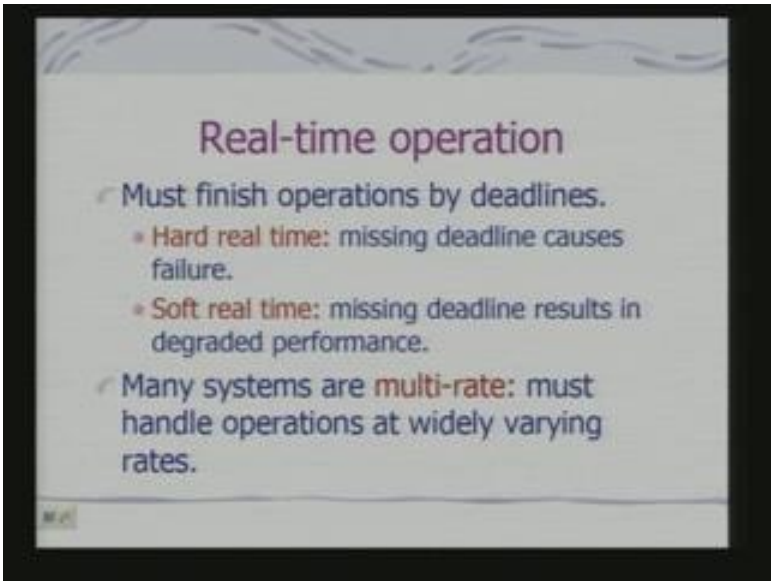
Say, for example, if I am designing an automated system for an aircraft, I can invest money for its development, I can use highly sophisticated equipments, but the same flexibility is not with me when I am designing a cell phone, a low cost cell phone aiming to serve a mass market. So the best technology choice will therefore depend on the number of units we plan to produce.

Now, let us to come back to this issue of real time operation that we started with. What is the real time operation? The basic definition is that operations must be completed by deadlines. So I have a deadline, so a real time operation must be completed within deadline. We have two kinds of real time deadlines; hard real time deadlines and soft real time headlines and accordingly also, we classify real time systems. In a hard real time systems, we cannot really miss a deadline. If you are talking about an atomic reactor control, if I miss a deadline then there can be a catastrophe. On the other hand, for a soft

real time systems, we can at times miss deadline. See for example, when we are playing a video on a laptop even if we cannot decode a frame in time, nothing catastrophic happens, only it disturbs you viewing experience. Many systems are also multi-rate that means this embedded systems are receiving inputs from the external worlds and these inputs can come at different rates.

So they need to handle this different rate inputs and we call them therefore multi-rate systems.

(Refer Slide Time 11:10 min)



There also various application dependent requirements, in many cases, just take for example, an aircraft system with definitely need fault-tolerance also for medical equipments when we are monitoring a critical patient using an embedded system, we do need fault-tolerance and reliability. Further, the systems must be safe; systems must avoid physical or economic damage to person as well as property.

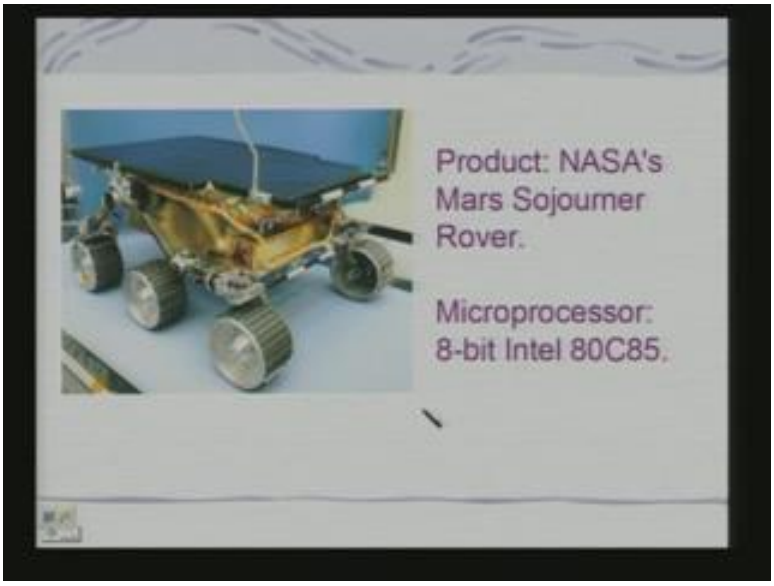
(Refer Slide Time 12:31)

Further, if they are dedicated systems, then the design consideration is obviously different because they are not expected to be programmed on a regular basis. So what we say, the programmability of this systems would be really used during the right lifetime of the system. That means once programmed this systems are expected to execute infinitely for a large duration of time without users intervention. And they are expected to be programmed or designed for specific tasks and therefore they are basically what we call dedicated systems. Let us try to look at some more examples; examples from the outside world.

(Refer Slide Time 13:41 min)

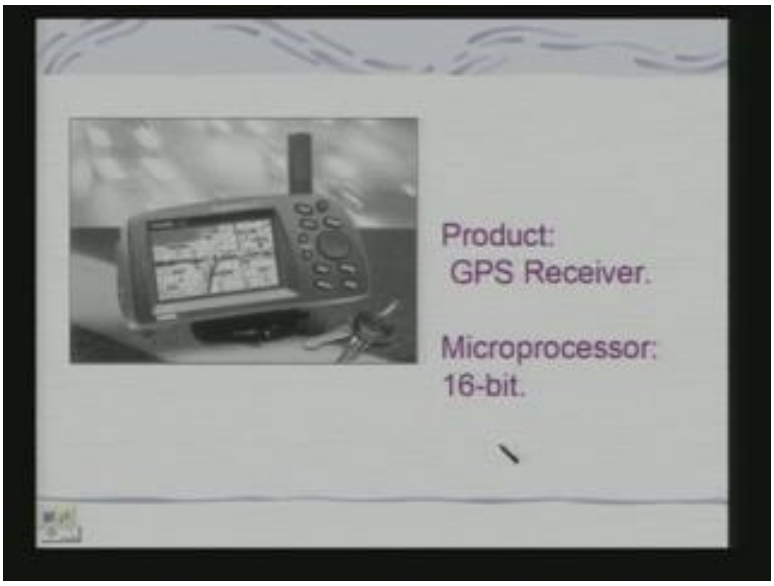
This is an example of a vending machine. We have seen vending machine at various points and they are all actually embedded systems. And in these embedded systems, it is not only the electronic part which is important, you can realize, but the mechanical part is also of critical important because you have to finally deliver the goods and accept the cash. This example is an old vending machine which uses 8-bit Motorola microcontroller. And this is a newer vending machine, okay, which is actually 2004 introduction product. This is a web enabled cashless vending machine. So, you can see that a simple task of delivering a good, in response to the cash input, now has been changed into a web enabled device. And what is the advantage? Because of the web enabling, the stock can be monitor remotely, the whole cash transactions can be through your credit cards or smart cards, as well as, the security also can be monitored from a remote location. This has happened again because you have brought in sophisticated processors, sophisticated functionalities on to this vending machine.

(Refer Slide Time 15:04 min)



This is another example; this is NASA's Mars Rover and this is an old robot, it is a mobile robot and it uses an 8-bit Intel microprocessor. In fact, this is a variant 80C85. It is a variant of 8085 microprocessor, which you may be familiar with and this is a robot which moved down Mars.

(Refer Slide Time 15:29 min)



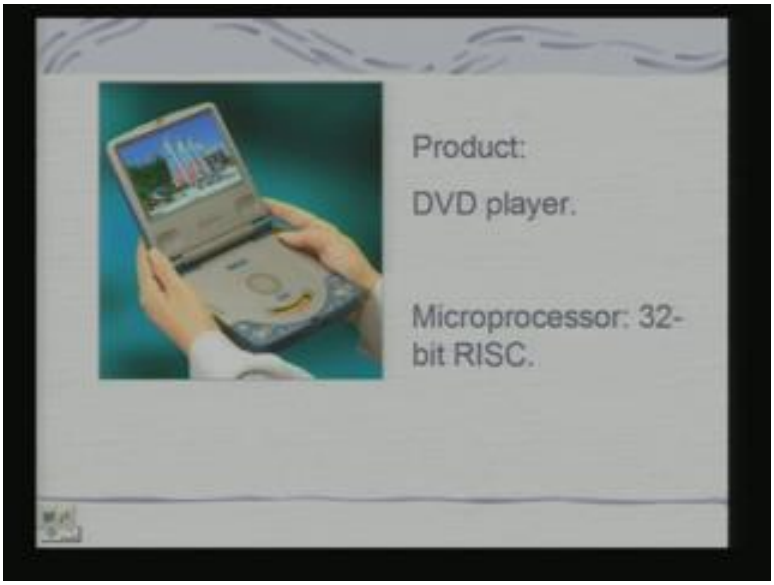
This is another product, is a GPS receiver, Global Positioning System which actually enables any transport vehicle to deprovement its location and for automotive systems which provides automated navigational tool, this GPS receivers are becoming a very common place. These are all embedded systems. Here a very critical component is the communication equipment; it has two receive input from the satellites as well as it has to provide output regarding its positions as well as its display, because display is critically important here when it is being used as a navigational tool. This is an MP3 player, various versions of MP3 players you are using, they are all embedded systems. What is MP3? MP3 is actually a what? A compressed form of audio and since we need to decompress audio to play, I need to do computational task, a pretty sophisticated computational task. That is why we will find that the microprocessor that is being used here. It is a 32-bit RISC microprocessor.

(Refer Slide Time 16:15 min)

This is another example of a DVD player; the same issue is applicable here. Why? Because your DVD has got video in a compressed form. So, I need to do decompression and decompression at what rate? At a video rate; video rate means what? 25 Hertz.

So, effectively of about 40 milliseconds to decompress a video file. So, I need also in this case, a pretty sophisticated microprocessor to work with. So I have got a 32-bit RISC microprocessor.

(Refer Slide time 16:58 min)



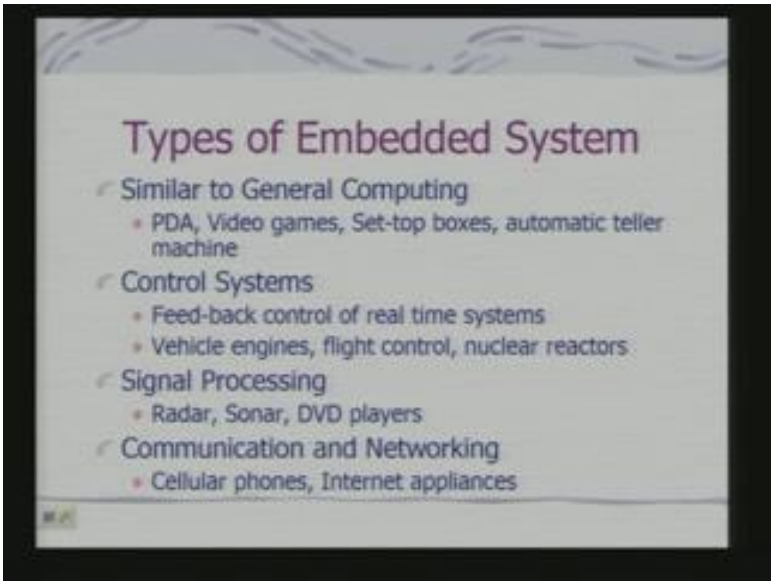
This is a Sony AIBO Robotic dog and it was very popular pet in Japan and it uses; just note it, this is the most complex microprocessor or a microcontroller that we have seen so far, it uses 64-bit MIPS processor. It uses 64-bit MIPS processor. Why? Because, it has to a handle of number of complex tasks. It has to coordinate its motions that mean its need to control the manipulator, it needs to do sensing, it also need to communicate. Because it also has the communication facility and if you are familiar with this RoboCup that is, the competition of Robo football between different robotics teams; in fact, this Sony AIBO robotics dog has been extensively used. And it has been built into various interesting algorithms into it to detect the ball, how to through the balls towards a goal. So, all these complex functionalities have been built into it so, it requires pretty sophisticated processors to handle it tasks. So, it is using 64-bit MIPS processor.

(Refer Slide Time 17:52 min)

So, now let us come to what are the different types of embedded system. We have seen variety of examples. Now, let us classify these examples into different types. Some are similar to general computing, like PDA, video games, set top boxes, automatic teller machines. Why they are similar to general computing? They are similar to general

computing simply because if you take PDA, the majority of the tasks that you do is a restricted form of the task that you do on a computer. Similar thing is with the video games, you provide the input; the user provides the input and it expects some output. They are not really sensing external environment on its own as well as they are not activating any actuator on its own that would influence or change the external world. So, these devices are more like a general purpose computing machines; they respond to users input. Others, on the others side have got control systems whose basic job is that of sensing and actuating. The feedback control of real time system, various real time systems, I need a feedback control depending on the external input, I need the control to take some actions. And examples of these are vehicles engine fuel injections to be controlled, flight control, nuclear reactors. These are all examples of embedded systems which belong to the category of control systems.

(Refer Slide Time 19:00 min)



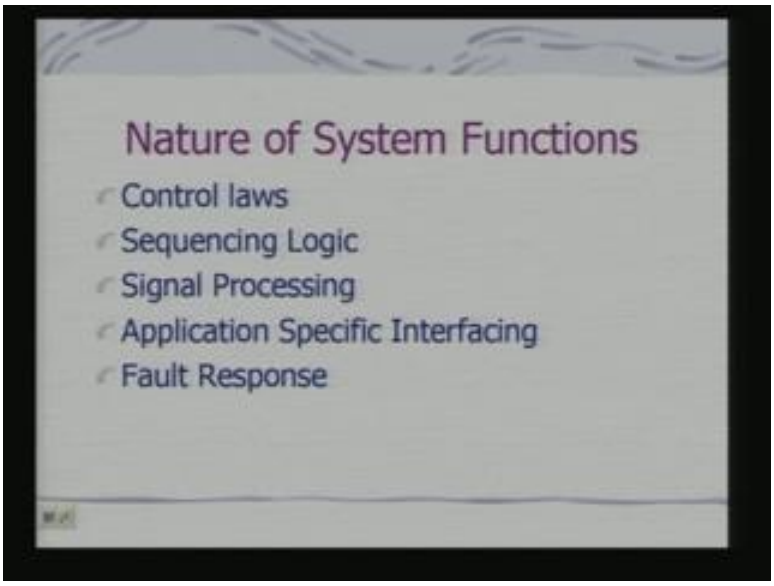
Next, we have signal processing because here the core job or basic focus is signal processing. Your MP3 players, your DVD players, radar control system; because in radar although there is a control system the basic job is processing of the data. Similarly, a sonar system; they are all example the signal processing systems. And communication

and networking is another category of which the most common example is your cellular phones. And now, we are getting a number of internet appliance, in fact, the web enabled vending machine is an example of this kind of an internet appliance. So what are the different kinds of functions that an embedded system is expected to implement?

First is, if it has got the actuation, sensing an actuation as a basic task it must realize some control law; it has to realize a control law. Second important issue is that there has to be a sequencing logic. This sequencing logic is obviously task specific and it is not a general purpose sequencing logic; it could have a task specific sequencing logic implemented into it. Third thing is that it should have signal processing if it is required and wherever and where we are interfacing an embedded system with external sensory input we need signal processing. So, in many cases, even when the signal processing is not core activity we need signal processing ability to deal with sensory inputs. Next thing is application specific interfacing because application will tell us what kind of sensors and what kind of actuators to be interconnected and accordingly we should have that interfacing. This interfacing implies both hardware as well as software.

Next thing is fault response; what happens when a fault occurs. A basic issue or basic design philosophy for fault response is what we know as, what we call graceful degradation. Catastrophic failure should not happen. The system should tell users that things are failing and gracefully it would degrade. Say, for example battery failure, there should be a message to the user saying that battery is low. So user can take some action. It should not suddenly stop its activity all of a sudden. So graceful degradation is another important function which is to be implemented.

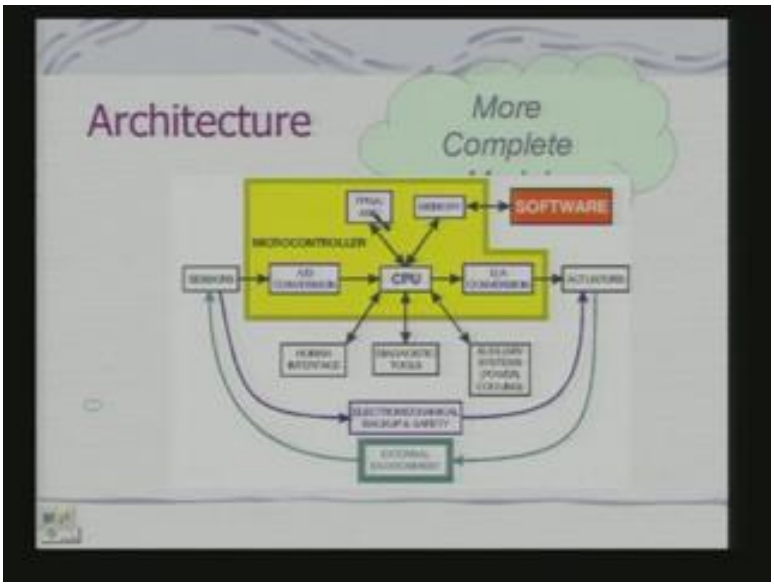
(Refer Slide Time 21:25 min)



So, let us look at now, a more complete architecture of the embedded system. We have seen the simplest model and now we shall make it more complex because we have now understood; what are its requirements, we have also reviewed some of the examples, so now, let us look at a more complex example. So, what are the things which are involved here? What I have shown if you go back to the previous model; I have now expanded the basic block. I have expanded the basic block and have added something more to the basic block. In the basic block, earlier I had just shown the CPU, along with CPU now we are showing obviously the memory because memory will have the software to control the system. Also we are showing analog to digital converters and digital to analog converters. This AD conversion blocks actually provides an interface to the sensor here and DA conversion block actually provides interface to the actuators, because an embedded system which is situated in an environment is expected to receive sensor inputs and actuate the actuators to change the external environment.

So, these two are very essential and integral component in majority of your embedded systems. Here I have shown an FPGA or ACIC block. Why? Because in many cases; my CPU may not have the ability to execute my software, satisfying real time constraints. Under those circumstances I might need special hardware to come or interfaced with my CPU. So that can be implemented on an FPGA and ACIC can be used along with the CPU.

(Refer Slide Time 25:02 min)



Now, let us look at other issues; one is obviously, on the CPU we need to implement, with the CPU with human interface; if you are talking about any kind of reading to be obtained through the embedded systems, any control functions to be altered by the human users I need a human interface. So, this interface becomes an important component. So you will find in many cases, you have an LCD display panel or a simple LED based informative colour codes by which the user can be informed of what is happening inside. Also there are diagnostic tools, why diagnostic tools? Because this, although this systems are expected to work forever there are obviously probabilities of failure and if a failure occurs, how to trace that failure? Can it be repaired or simply it has to be taken out and thrown away? So, if it has to be repaired I need to have diagnostic tools to interface and check whether it is working. Second important thing why diagnostic tools are important; that when the system is starting up or system is working on, even on a continuous basis, it should do some self check to know whether all the parts of it is functioning properly or not. Because if all the parts are not functioning properly, what can happen? It can actually do damage to the users, because of malfunctioning of some hardware components. So, it is also expected to do some self checks at regular basics. So, diagnostics tools form also an essential component and you have the auxiliary systems which are to be dealt with

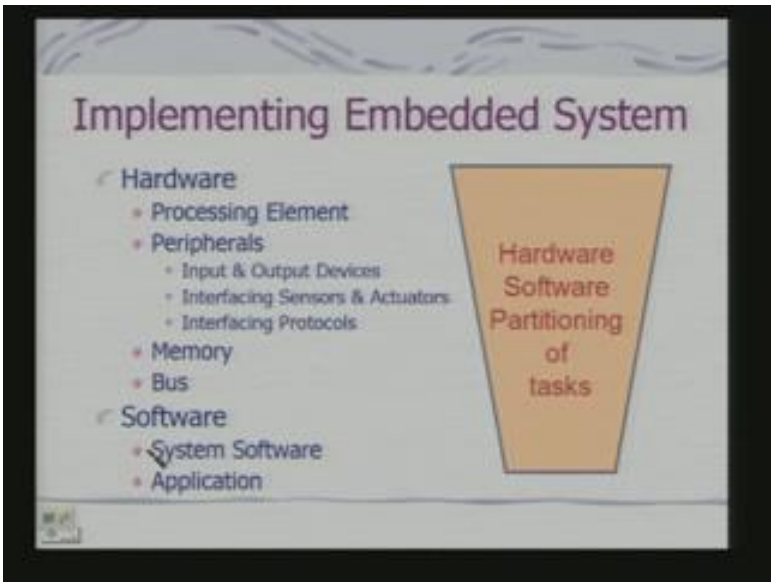
power, because if there is a power dissipation, then cooling becomes an essential component.

So, how to design the cooling circuit, how to take care of extra heat dissipations and this mechanical aspects of this design becomes important. Obviously the casing; the casing the whole system should be properly packaged. If it is not properly packaged and the packaging should be as per the requirements of the external environment in which the embedded system is expected to be pleased. So, this packaging becomes a very-very important issue and in fact, if you look at this packaging issue; this if your packaging is not properly designed even a good well design system can fail. Because then, because of the bad packaging the system can get, say for example, moisture. That moisture can go in and affect the electronics, then heat can affect electronics, so all these mechanical aspect of the design become extremely important. Although, in the course we shall not discuss those mechanical aspects of an embedded system design. But, please be conscious about the fact that these aspects are very-very important for any kind an appliances design and implementation.

So now, if you know these as an architecture, so how to implement an embedded system? And this is exactly, will be the focus of our course. We shall learn more about what is presented in the slide. We shall discuss obviously the processing elements. The processing elements are basically your microprocessors and microcontrollers. We shall look at the peripheral devices because input and output devices become a critical component in this context. And also how to interface sensors and actuators and there also there are various kinds of interfacing protocols which can vary from one sensor to another sensor. Then you have got memory, also the bus design. So these are different aspects of an hardware of an embedded system and if you look into it, these aspects of the hardware are very-very similar to a general purpose computer. There is no basic difference conceptually from that of a general purpose computer. The only issue which is of importance is, these aspects: in a general purpose computer we tend to talk about standard input output devices although that set is getting expanded day by day, but we tend to talk about standard input output devices. But here, the set of input output devices are large because there can be different kinds of sensors and each sensor has got it own characteristics. And therefore, I need to, you will find that particularly the processors

which are targeted for embedded applications will have very sophisticated and a complex mechanisms; in many cases even simpler mechanisms not only complex mechanisms to interface with external devices and external IO devices in particular.

(Refer Slide Time 31:06 min)



Then we come to the software, here also you will find that I have talked about system software and application software which is again very-very similar what we have for a general purpose computing needs. But here, the system software has got various components. One aspect, obviously all of us know what systems software's are; what are system software's? Typically we talk about assemblers, compilers that is language translators are a class of system software. The other class of system software are operating systems. Now, in majority of the cases, embedded systems will have specialized operating systems and not general purpose operating system like your windows, units or variance of them. There would be specialized operating system because they should satisfy certain characteristics of this embedded system. And the most important characteristics is what? The OS which you encounter in general purpose computing systems, they have been designed to satisfy the general purpose need; the requirements of a programming for various needs and tasks. But, in this case, these embedded systems are dedicated, so its OS are also tuned for that kind of a requirement.

They also have the real time scheduling features because in many cases we require real time scheduling.

But, apart from these OS, in a general purpose computer, you also compilers; you have compilers which compile your high level language code to the target machine code to be executed on that system. But, in case of an embedded systems, you will find what we call cross assemblers and cross compilers and various kinds of other development tools. Because its entire development process, that is your high level language program specifying the software, would take place and on a host system and not exactly on the target processor. After you have tested, may be the software and everything, the software will be loaded onto the target system. So, you get cross compilers and cross assemblers.

What is a cross compiler and cross assembler? Here, say for example, if I am using a compiler for PIC microcontroller, so that compiler would run maybe on a simple PC in a windows environment. But, so you write a C program, you use that compiler to compile your C program and what it will generate? It will generate the code for PIC microcontroller and that code has to be loaded on to your target board which has got the PIC microcontroller and it would get executed on that target board. It will not be executed on the PC in which your compiler is running. So, this is an example of a cross compiler.

Also, you will find this compilers and assemblers have got various interesting features. You have got compilers for a family of processors that means it is just not targeting one processor but it is targeting variance of this processor. Because these processors have got very similar architecture; there maybe some differences in the number of registers etc and that can be taken care of by the compiler by the appropriate input. Also, there are tools which also come as part of system software which are called emulators.

What are emulators? You have got instruction set emulators. This instruction set emulators actually emulates your processor on another target machine. There can be simple behavourial emulator that is, it just emulates the behavior of the target processor; that means, it simply implements the instruction set of the target processor. In many cases, you can also have a complete simulation environment where you can even do a timing analysis of your code on a host machine. So, these are the tools; system software tools which are typically targeted for embedded system development.

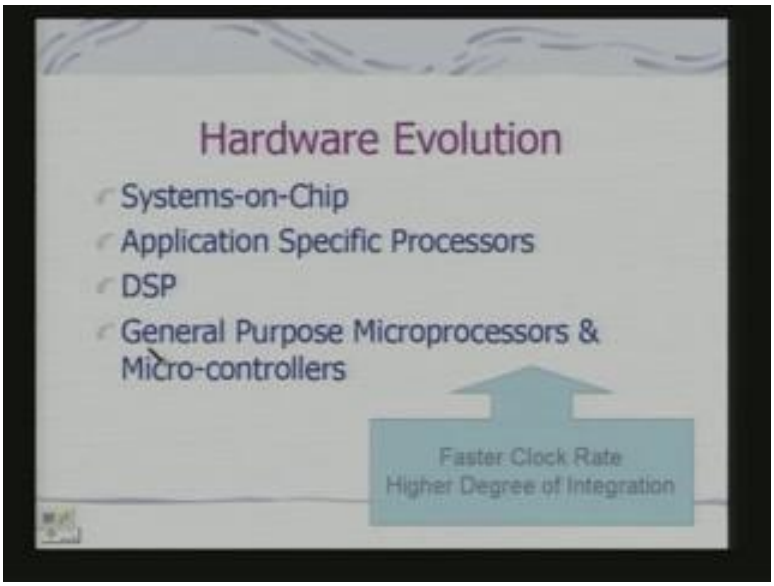
There are also other tools were you have got actually combination of the two that means you have got some hardware as well as some software. That means, say for example, you have got a target board. So, on a target board there will be a simple software to execute your code and there will be another layer running on the PC. Because the PC will be connected to the target board through a hardware connector, the code that you have developed on the target board can be loaded via the connector on to the target board. And then what will happen? You can monitor execution of the code from the PC itself on the target code; so these are debugging tools. So, to summarize, if you are talking about system software, we are talking about what; compilers, in particular cross compilers and cross assemblers. We do talk about emulators and simulators and we talk about debugging tools. So, this system software set is obviously different from that what you expect for general purpose computing needs. On top of that, we have OS, the operating systems which are targeted for dedicated appliances and many times they do support real time scheduling capabilities.

The next thing is application software's; obviously application software's give the flavors, different kinds of flavors to the different devices although they may support the same system software. Take, for example, you may have the same operating system vagues works running on your laser printer as well as maybe running on some other appliance. But, application software on the laser printer is targeted for printing and it is supported on top of an OS which is targeted for embedded system. Although that OS can be present in multiple such appliances, but your application software would distinguish the functionality of these appliances. Let us look at the history of hardware evolution because that also have led to this status; today's status of embedded systems.

At the lowest end I have got general purpose microprocessors and microcontrollers and in fact, this arrow actually tells you which one of these cases you have got; with the time what does happened is you have got a faster clock rate and that means what, you have got faster execution speed also you have got more higher degree of integration that means more and more devices and peripherals have got integrated into the chip. And the general purpose microprocessor microcontrollers, what is the advantage of them? You can get them off the self and using them you can develop a system, so NRE cost towards

development of the processor is minimized when you are using general purpose microprocessors and microcontrollers for implementing an embedded system.

(Refer Slide Time 37:55 min)



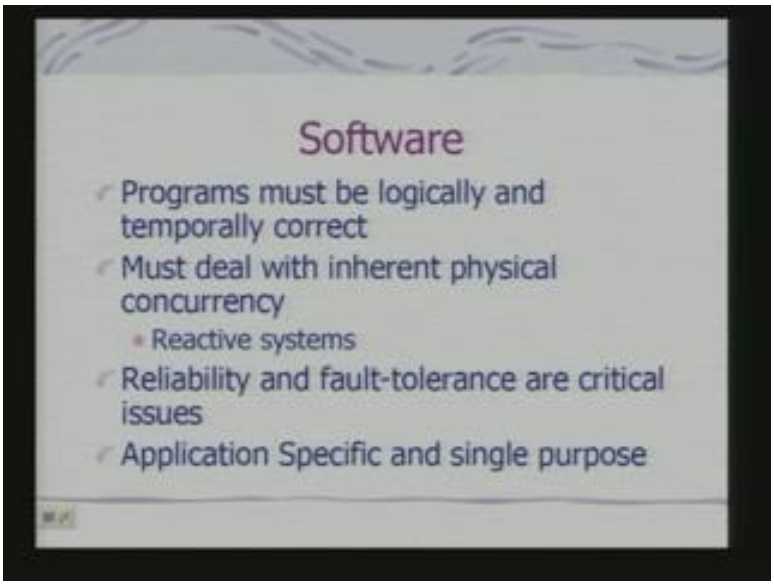
DSP, that is Digital Signal Processors, I am not talking about digital signal processing, I am talking about the Digital Signal Processors. They are particularly required when your basic task is that of signal processing and there are variety of signal processor with different architectures which are today available. But, obviously the cost of, when you are using a DSP will be more than that of a general purpose microcontrollers in many cases. It is not although universally true. Then we have got application specific processors, here what is happening is, you are trying to look at designing a processor which may exactly suit your application need. So, in this case, obviously the NRE cost is more and your application is such, that you can permit this additional NRE cost. At the end of this list, that is on top I have put System-on-Chip, SOC's; and SOC's are current trends and you will find in an SOC not only a single processor code, but multiple processor codes along with peripherals are getting integrated. Why? Because today, you have the ability to do higher degree of integration. An example SOC is a Texas Instruments OMAP Processor which has got an arm which is a risk processor as well as a TI DSP sitting inside the chip

and they are and the entire communication and other peripherals are also integrated into the same chip.

So, currently you will find, there are variety of SOC's available; the system on chips. And you can understand, when we are talking about the system, it means not only a single processor and its peripherals but also a large number of peripherals along with even a special purpose coprocessors and even multiple processors being integrated together onto a single piece of Silicon. So, that makes what; if I have that, that means I can a much more sophisticated functionality being implemented into an embedded system as single Silicon would actually mean a smaller area. And in many cases, this associates a design in a power optimized fashion and hence less consumption of power.

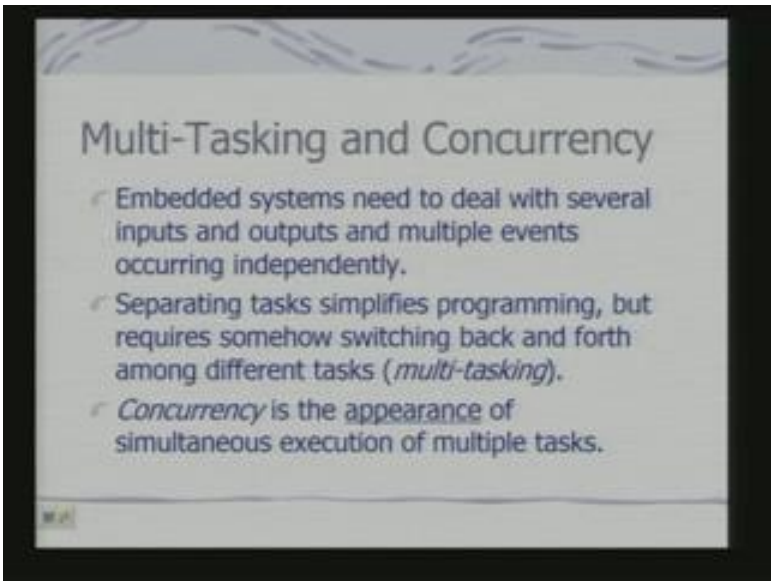
Software; what are the typical characteristics of the application software and the operating systems that supports, that means in an execution time what are the typical characteristics they should have? The programs must be logically and temporally correct; logically correctness we all understand, but the most important thing is temporal correctness, in this case, when we have real time consideration. Because, I cannot do something correct at wrong time, then the correctness has no meaning. Obviously, they must deal with inherent physical concurrency. In a general purpose computer, we talk about concurrency simply because there may be a multiple users, multiple processors running. Here, physically, since the world is concurrent, I have to support concurrency and along with it reliability and fault tolerance; obviously critical issues. And here, what we are trying to refer to is fault tolerance and reliability not only of software, not only of hardware but that of software as well. And obviously the software has to be application specific and single purpose.

(Refer Slide Time 41:48 min)



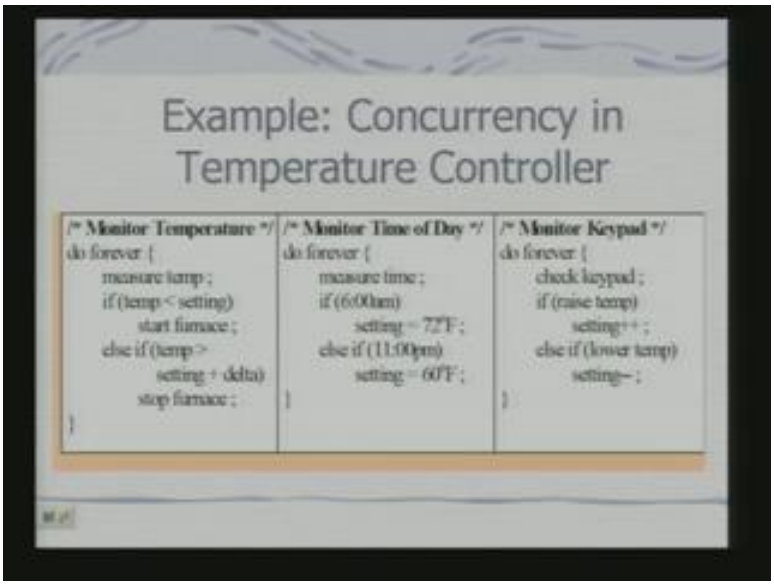
Let us look at this multitasking and concurrency. We are all familiar with this definition, this is just review. So, why multitasking is important? For embedded systems need to deal with several inputs and outputs and multiple events can occur independently. So, an embedded system in many cases, as expected to be multitasking. And separating task, another issue is separating task, simplifies your programming complexity. But obviously if you have a multitasking system, we need a kind of an OS kernel which would support switching back and forth, that is switching of the processor between different tasks; and concurrency is basically appearance of simultaneous execution of multiple tasks.

(Refer Slide Time 42:26 min)



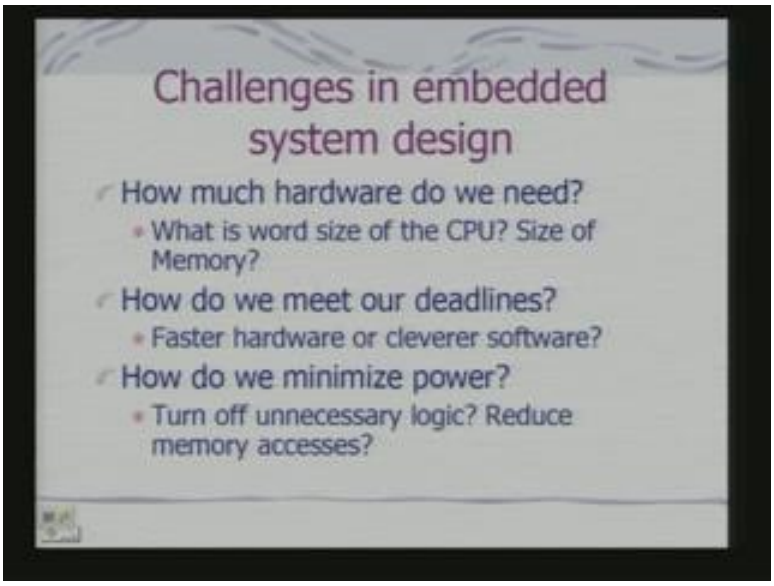
So, let us take an example; this is an example of a concurrency in temperature controller. It is a simple temperature controller on a furnace. And, you can say that it is supposed to just control the temperature, why should it handle concurrency. Just see why it requires to handle concurrency. Obviously it is monitoring temperature and depending on the temperature it is doing some settings. But, there are other issues which can come with it because depending on the time of the day the different temperature setting can be specified. Also, the user can do some modification in the setting from the keypad. So, effectively these are three concurrent events that can occur and these have been separated into three concurrent processors or tasks and being handled independently. So, a very simple embedded system also requires concurrency because the external world interact with the system in a concurrent fashion.

(Refer Slide Time 43:26 min)



So, that is why concurrency becomes a very important issue in this context, in just not having multiple processors from multiple users being run on a general purpose system. So, therefore what are the challenges in designing an embedded system? First is, how much hardware do we need? What is the word size of the CPU? What is the size of memory? It would definitely depend on what is the task that you are trying to handle. Then how do you meet our deadlines? This one deadline is not project deadlines, but deadlines to be met for a real time system; faster hardware or cleverer software and in fact, there may be cases I might write a clever software but it might not still meet my deadline on the CPU as it gets executed. I might require a faster CPU, but faster CPU can mean extra cost. So, what do I do? I try to get a compromise. What can I do? I may design on an FPGA, a dedicated function. So, I use a low cost CPU, but that function for which, I cannot meet the deadline using the software I design a dedicated logic on an FPGA or make it into an ASIC and include that with my general purpose micro controller. So, this is a very important point when we are dealing with real time systems.

(Refer Slide Time 46:13 min)



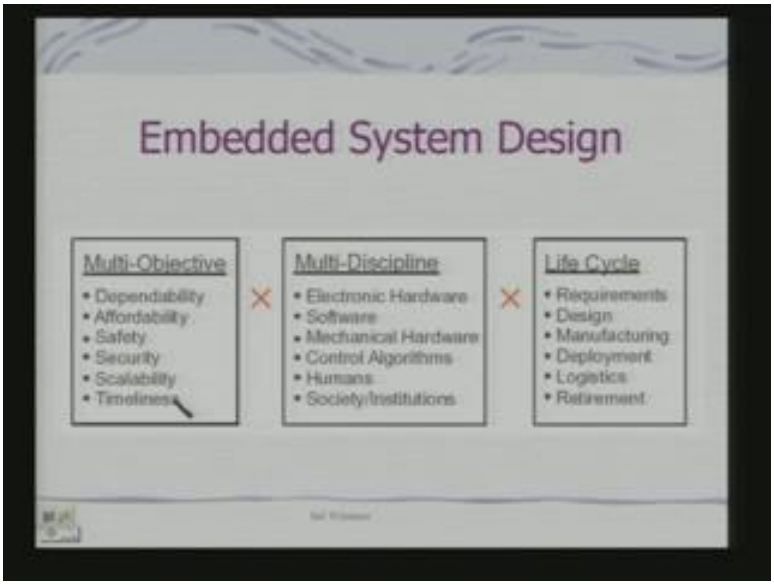
Next issue is how do you minimize power? Turn of unnecessary logic, reduce memory access; reducing memory access. Why? Each memory access will lead to consumption of power. When we discuss this power management issue later on, we shall see why these issues come up. So, this becomes a very important point to deal with when we are designing a system. See, if you look into it, the global picture for an embedded system design; just look at the themes which you are involved. It is a multi objective. Why? Because we have just tried to list some of these objectives: dependability, affordability, safety, security, scalability, timeliness.

We have already discussed the timeliness as an issue because I have to do computation in time, it has to be dependable; so it should not fail arbitrarily, it should have some kind of fault tolerance and graceful degradation. It should be safe and secure it should not cause bodily harm to the users and depending on market that we are looking at, it should be affordable. Therefore, if these are the objectives, in order to mid the objectives we require a kind of a multi disciplinary approach. Why? One aspect is, electronic hardware, the other aspect is mechanical hardware we have already talked about. The control algorithm is something absolutely important; the other thing is human and society or institutions. The sociological aspect about accepting a product; you can make a product but people may not accept it because it is not sociologically acceptable. Depending on

norms of the society; so the sociological perspective for introducing an appliance is very-very important.

And these are the different life cycle events; that is what you mean by life cycle? How the embedded system gets developed. I need to do requirements, then I need to do a design, look into manufacturing, look into its deployment, look into its logistics of maintaining the systems and then retirement means how to withdraw that product. Because after introducing a product, you cannot simply say that I won't support that product because you are a consumer, you have invested money into it and you have to support; there is a commitment to that product. So, the retirement plan of the product is also important.

(Refer Slide Time 46:54 min)



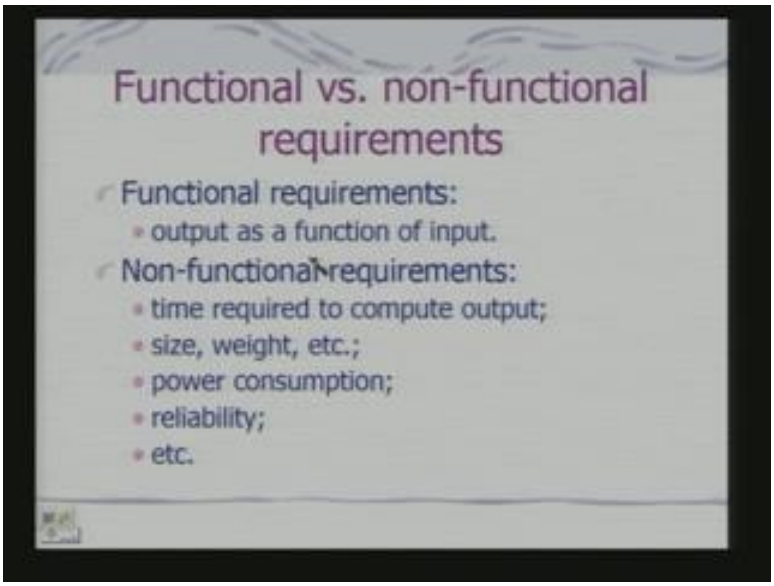
So, the design objective is, if you look into it, so we have got a very important design goal in terms of performance, the overall speed and deadlines. Then functionality and user interface, manufacturing cost, power consumption, physical size; these are very-very important. Look into it because these are; these are not always obvious I may just give you the performance as a criteria, but your weight and power consumption although related to this, have to be satisfied. Otherwise your product will not be acceptable in the

market. You cannot make a digital camera which would weigh may be 10 KGs, nobody will buy it.

(Refer Slide Time 48:45 min)

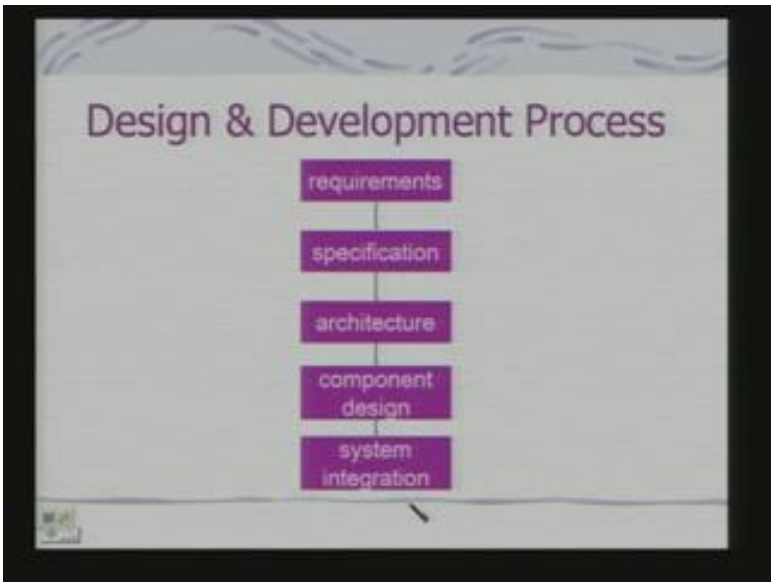
So, we talk about therefore functional and non-functional requirements. Why? Because functional requirement is what is output as a function of input; that is how you specify embedded system. And what are the non-functional requirements? Non-functional requirements are time, size, power consumption, reliability etc and these also should therefore form part of your design goal and design objective. I cannot ignore them because non-functional requirement at times are very-very important for acceptance of an appliance or an embedded system.

(Refer Slide Time 49:35 min)



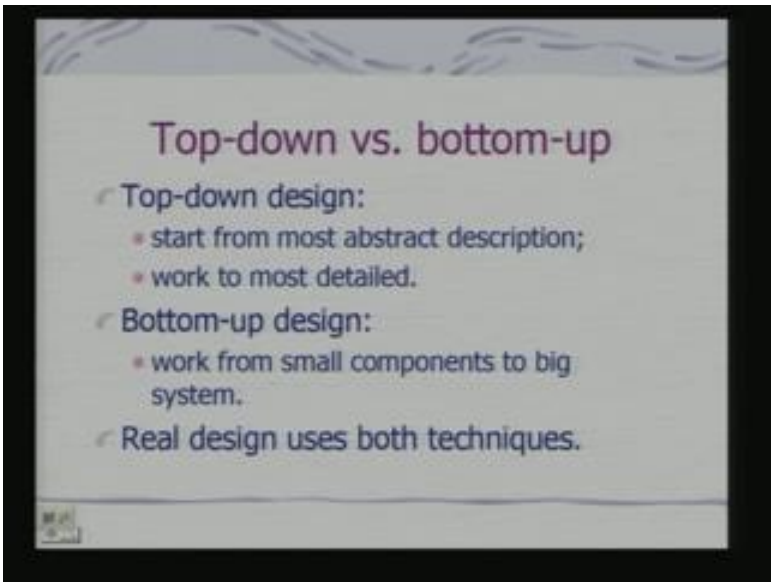
So, this design development process; that the life cycle that I was talking about. So, I start with requirements; builds up a specification, then go through the architecture, that is design an architecture. That is, this architecture is a block level architecture; then you look at component level design, then you do look at system integration and what I have not yet shown is basically testing face. Because testing is a very-very important component for an embedded system; because it is know that for an OS on a general purpose computing, if you find a bug, later on you can download a patch. Fine. And you can rectify that bug in the OS. Say, for example, but in case of an embedded system that flexibility is not with you. You are giving that product to the user and user is expected to use for ever and user is also not expected to be a computer savvy that he will connect to the internet and download the patch. So, these systems have to be very carefully tested and debugged for hardware as well as software faults.

(Refer Slide Time 50:33 min)



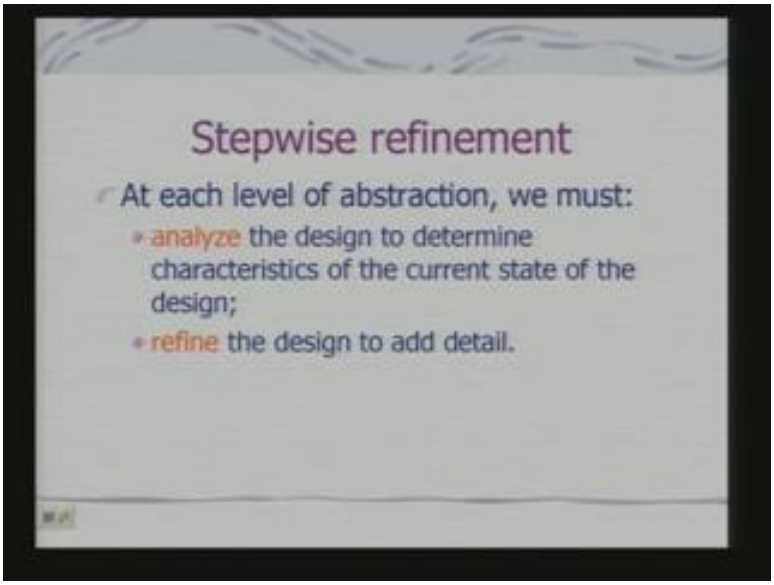
The design approaches can be top-down or bottom-up; just like any software design. Here also this issue comes up. In a top-down design, you start from the most abstract description and work down to the most detailed level. The bottom-up design which is also very-very common in terms of embedded system design strategies; you work from small component to big system. Because, in many cases, when somebody is developing a product, you have got parts of it already developed with you. May be some of the components from previous system, because it is available with you; you would like to use this component. So, you try to go through a bottom up process and in fact any real design actually involves both.

(Refer Slide Time 51:31 min)



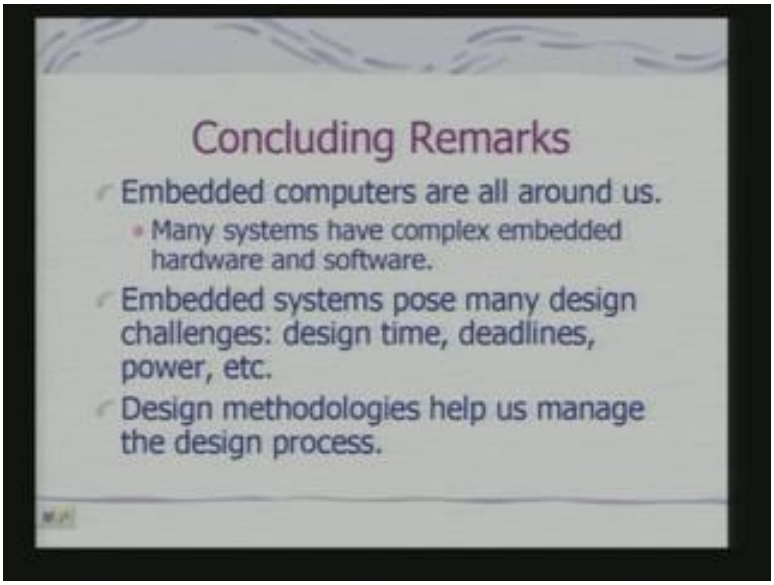
The other important issue is that of a stepwise refinement. Whatever we talked, you know, about the software development here, is equally applicable and here we not just talking about software development but we are talking about both hardware as well as the software development. And you have realized, I think, by now that this hardware and software development for an embedded system go hand in hand. I cannot really separate things out because you have seen that, if some; if I cannot meet a deadline by using the pure software on a general purpose microcontroller, I might need to design a special purpose hardware. And in fact, that is exactly leads as to what is known as software hardware; who designing software hardware partitioning and those approaches.

(Refer Slide Time 52:59 min)



So, the stepwise refinement what we have talking about is, stepwise refinement of both hardware as well as sort of the software and what does that mean, stepwise refinement of the system as a whole. So, therefore we come to this concluding remark because what we have so far covered is a broad overview; an introductory overview of what is an embedded system. So, what we think we have seen is; we have got various appliances which are embedded systems. In fact, somebody made this statement that today we have more microcontrollers and microprocessors at home than computers all around us. It is absolutely true, because you have your washing machine, you have your microwave, you have your cell phone, you have your TV, you have your PDS; everything have an embedded system, everything has got a microprocessor or a microcontroller setting inside and you are using it. So, embedded computers and embedded systems are everywhere and we need to know how to design them that is the basic issue.

(Refer Slide Time 54:26 min)



And therefore, embedded systems pose many design challenges: design time deadlines and power and that is precisely the reason why you need to deal with it in a specific way. And you have also realized that embedded systems, in what way are different from a general purpose computer. Although the basic principles are very similar, but in many ways it is different. So, our design methodology and the principles which goes into design, as well as, characteristics of components which are used an embedded systems are expected to be different. And, we have seen also the design methodologies help us mange the design process far better. Do you have any questions? If you don't have questions, so we shall end this lecture here. In the next class, we shall start our discussions on embedded hardware in particular processors.