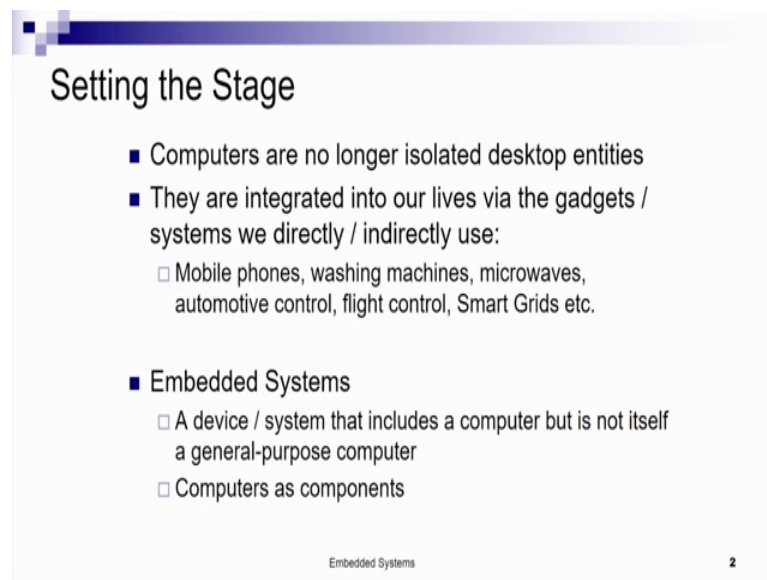


**Embedded System-Design Verification and Test**  
**Dr. Santosh Biswas**  
**Prof. Jatindra Kumar Deka**  
**Dr. Arnab Sarkar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Guwahati**

**Lecture - 01**  
**Introduction**

Welcome to the first lecture on embedded systems design.

(Refer Slide Time: 00:34)



**Setting the Stage**

- Computers are no longer isolated desktop entities
- They are integrated into our lives via the gadgets / systems we directly / indirectly use:
  - Mobile phones, washing machines, microwaves, automotive control, flight control, Smart Grids etc.
- Embedded Systems
  - A device / system that includes a computer but is not itself a general-purpose computer
  - Computers as components

Embedded Systems 2

to start let us define the premise of what we are going to study in this course. We are going to study embedded systems. So, computers are no longer isolated desktop entities with a monitor with a CPU and a keyboard only. They have been they have got integrated into our lives via various gadgets and systems that we directly or indirectly use. For example, microwaves, washing machines, mobile phones, automotive control systems, our cars, flight control systems, the smart power grids etcetera.

So, what are these computing devices? What are these what are the characteristics of these systems? These are embedded systems are embedded computing systems. An embedded system is a device or system that includes a computer, but is not itself a general purpose computer. So, therefore, you have a computing capability within which is embedded within your microwave, but your microwave does something else the

computer is inside, which takes your programs as to how to cook your food, washing machines can operate in various ports.

So, therefore, you have a computing platform inside the washing machines as well. So, therefore, these are computers as components within bigger physical systems.

Now, if we look at the examples that we showed here, we have seen essentially simple devices simple single devices like microwaves, washing machines, mobile phones to very complex devices like automotive control systems and flight control systems and even big geographically big distributed systems like the smart power grids, whereas smart power grid will have a generation side, a transmission side, a distribution side and consumption of electricity from generation to consumption of electricity.

And let us say, our computer within that embedded computer within computes at each point in time. What is the real time balance of the power? For example, if I am generating a certain amount of power and consuming another amount of power in real time, what is the balance in power that we have in the grid and ideally it should always remain balanced. So, those computers are come that are embedded within such big systems will also be called embedded systems. So, it is not only a single device, it can be embedded into a bigger distributed system as well.

(Refer Slide Time: 03:28)

### What's Important for Embedded Systems

- Stringent resource constraints
  - Compact systems
    - Simple processors
    - Limited memory
  - Reactive
    - High Performance
  - Real-time
    - Predictable
    - Ensured-worst case response
  - Fault-Tolerant
  - Low power
  - Time-to-market

Suppose you want to design a pacemaker

- Resource optimization → Reduce Cost
- Test automation and Design-for-Test → Improve Testability
- Algorithms for better yield → Improve Reliability
- Improve Verifiability → Verification reliability and coverage
- Minimize Power → Power optimization during synthesis
- Minimize Area → Area optimization during synthesis
- Minimize Delay → Delay optimization during synthesis

Embedded Systems 3

So, what are the typical characteristics of embedded computing systems? Let us consider this taking an example of a pacemaker; that is, let us have to be placed in a heart of a person. What does a pacemaker do? A pacemaker generates electrical impulses through its electrodes so as to compensate for missing pulses, which are generated naturally within the heart.

So, let us say the objective of our pacemaker would be to generate artificial heart impulses. Whenever, the natural sequence of impulses are not periodic enough or the frequency is not adequate enough, then our pacemaker has to come up and compensate for the impulses that could not be generated naturally.

So, therefore, what are the design constraints in this system? It has to have it has to minimize delay why? Because, we said that, whenever the heart misses, fails to generate impulses, the pacemaker should come in and generate impulses. So, therefore, I cannot because, this is safety critical. I cannot say that, I got delayed in generating an electrical impulse. I should be always be able to generate impulse within a given certain interval of time. So, if it is missing an impulse within a certain period of time, I should be able to generate the artificial impulse. So, it should minimize delay.

It should minimize area the computing platform, should minimize area why because, it has to be a small device placed within the heart. So, it cannot be big it has to minimize power. So, these pacemakers should remain in your heart at least for say 10 20 years. So, the battery has to be capable enough to run for 10 15 years, 20 years. Hence, the consumption of power by the computing system has to be low. It should improve verifiability meaning that, the design of the system is safety critical.

So, we should be able to verify that the design meets all the specifications requirements that we have put up before the design. So, I have certain properties the design should meet, certain safety properties, certain functional properties, energy etcetera all these properties is the design meeting. So, I my design should allow the verifiability and coverage of all important properties that the system should satisfy.

We should have algorithms for better yield. So, how can we have algorithms? It should be fault tolerant as well. The system should be fault tolerant as well. The design should allow late testability of the manufactured system. So, I should have certain points in the

in the system through which ports in the system through which I can test that the implemented system the manufactured system is working correctly.

So, it should allow design for testability and last but most important, it should be low in cost, it should be available to the masses. So, the manufacturer product cannot be extremely costly. Hence, all these design constraints need to be taken into account when we are designing an embedded system such as a pacemaker.

Therefore, if we on the left, I have jotted down a few on the left, I have jotted down a few typical important factors that an embedded design has to take care. For example, the computers the embedded computing systems that we use may have stringent resource constraints. They may be compact systems, may be some single simple processors because, again I said it has to minimize area, it has to minimize power.

So, it cannot be very complicated. It may many a times it may have simple processors limited memory within which I have to meet the performance criteria such as minimize delay as well. It has to be reactive meaning, high performance minimize delay, high performance. So many embedded systems need high performance.

It has to be real time meaning that it has to be predictable. So, when I it has to be able to say that when the heart misses an impulse at say, time  $x$  within time  $x$  plus 10, I should always be able to generate an artificial impulse from the pacemaker. Therefore, it has to be real time meaning that, it has to be predictable. It should has ensured worst case response times, it should be fault tolerant, it is a safety critical device placed within the heart.

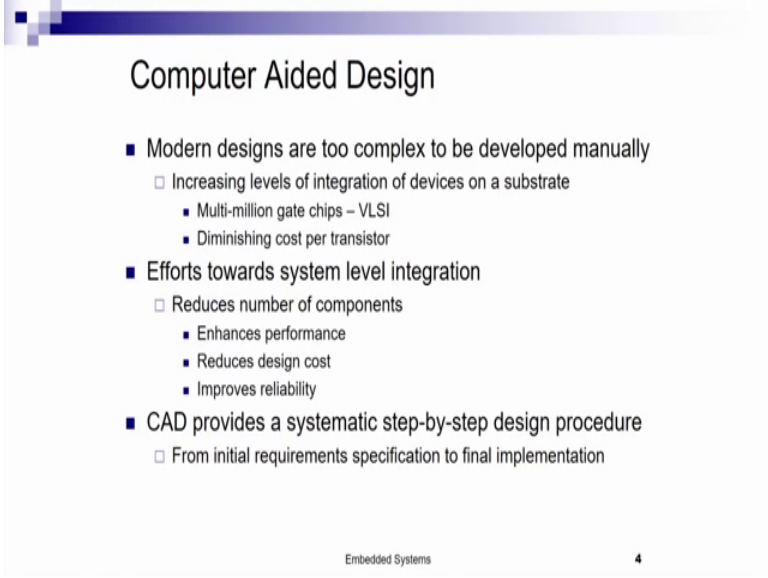
So, one module malfunctions. Then, another module at least there has to be a failsafe state, where maybe even at reduced performance. It performs, it is able to function, it does not completely crash. So, it should be tolerant to faults in the different modules of the system.

It should consume low power as we talked up for the pacemaker and it should have short time to market. So many for many embedded systems the time to market becomes very short. Because, you we have discussed discovered an use case for a certain embedded device.

For example, let us say, a new type of digital camera we want to design and put in the market. So, if I take a long time to design and bring it to the market, the need for the device may go, may not exist anymore. So, therefore, it has to have quick time to market as well. So, these are all different design issues that are related to embedded systems.

Now, to satisfy all these design issues, the design has to be automated. I cannot do a hand design of complex embedded systems and be able to satisfy all the different issues that we talked about in the last slide, hence, we require computer aided design.

(Refer Slide Time: 10:07)



**Computer Aided Design**

- Modern designs are too complex to be developed manually
  - Increasing levels of integration of devices on a substrate
    - Multi-million gate chips – VLSI
    - Diminishing cost per transistor
- Efforts towards system level integration
  - Reduces number of components
    - Enhances performance
    - Reduces design cost
    - Improves reliability
- CAD provides a systematic step-by-step design procedure
  - From initial requirements specification to final implementation

Embedded Systems 4

So, modern designs are too complex to be developed manually increasing levels of integration of devices on a substrate. Now, we have platforms or chips which contains multimillion gates, where extreme high levels of integration within the chip and therefore, that we have diminishing cost of transistors. So, the amount of functionality that we can put into a chip is increasing and therefore, more and more complex embedded systems are being designed and these complex embedded systems are difficult to design manually.

We have higher efforts towards system level integration. Now, we are designing a SOCS. For example, our mobile phones contain processing units processing platforms that contain say our general purpose processors, special purpose processors, power management units, RFID. So, some are analog, some are digital and they are all

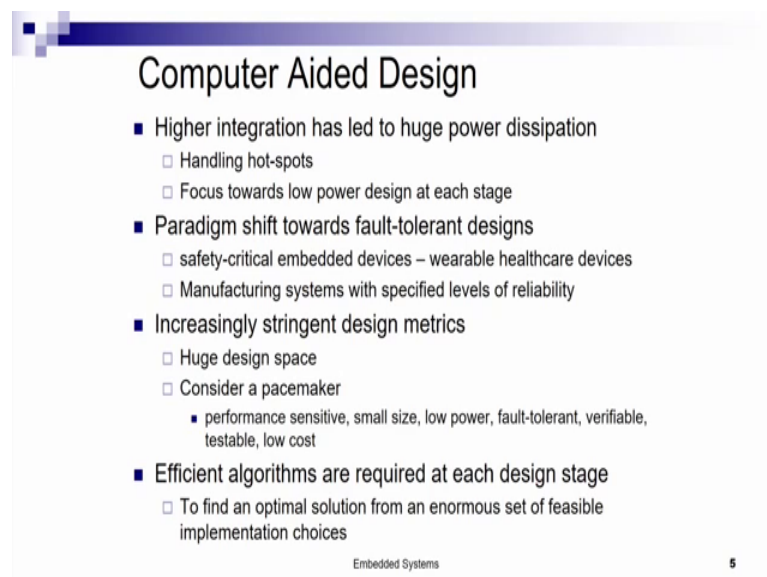
integrated on to the same substrate. So, they are on to the same board. Hence, we are talking about systems on chips systems on boards the.

Why do we have higher levels of integration? Because, high levels of integration reduces the number of components and such reduction in the number of components enhances performance because, our component to component communication delay will reduce the failures. if you have a components in the same module or the same chip, the chances of failure are also reduced during let us say the communication.

It reduces design costs, it improves reliability, CAD provides a systematic step by step design procedure. Another reason why computer aided design is used is that, because it allows a systematic step by step design procedure, handmade designs can be Adhoc. We may think that we are doing a very good design, but if it is not systematic and step by step, our design process may not produce optimal results. Hence, we need CAD which provides a systematic step by step design procedure from initial requirements specification to the final implementation.

So, first, before the embedded system is designed, we only have initial English language. Let us say, these are the requirements that my device should satisfy. These are the kinds of requirements specification; we have these are hazy it is not formally defined from that English language, incomplete specification. We need to formally step by step computer aided design steps take us to a formal modelling design and implementation.

(Refer Slide Time: 13:02)



## Computer Aided Design

- Higher integration has led to huge power dissipation
  - Handling hot-spots
  - Focus towards low power design at each stage
- Paradigm shift towards fault-tolerant designs
  - safety-critical embedded devices – wearable healthcare devices
  - Manufacturing systems with specified levels of reliability
- Increasingly stringent design metrics
  - Huge design space
  - Consider a pacemaker
    - performance sensitive, small size, low power, fault-tolerant, verifiable, testable, low cost
- Efficient algorithms are required at each design stage
  - To find an optimal solution from an enormous set of feasible implementation choices

Embedded Systems 5

Higher integration has led to huge power dissipation, because we have multimillion gates for chip we have higher integration on the same chip. So, power dissipation may be for example, through leakage, through the transistors, through the many transistors in this highly integrated system has increased it is generating hotspots which because, leakage power generates high temperature and higher temporary, higher chip temperatures may induce faults in the system which are called hotspots.

So, if at a particular point in the chip the temperature rises, suddenly you create hotspots. So, the design should take care that hotspots are not there and this may not be possible through handmade designs. So, computer aided design again helps us there focus towards low power design at each stage. So, that can be done only through systematic computer aided design.

There is a distinct paradigm shift towards fault tolerant designs. Even simple devices for example, our pen drives. Now, come with now, now are starting to come with certain quantifiable levels of reliability, say it will run for these many these many reads and writes ok. So, safety critical embedded systems such as variable healthcare devices, all these things even simple designs as I said are coming with quantifiable levels of reliability and hence these designs need to be fault tolerant.

Increasingly, stringent design matrix as we talked about the space pacemaker that is performance sensitive, small in size, low power, fault tolerant, verifiable, testable, low cost. So, all these design constraints are very stringent to build to bring together.

So, we need designs that satisfy all these constraints which is very stringent to meet and handmade designs may not be sufficient to do all this. So, therefore, we need computer aided design again and we need efficient algorithms at each stage of the design because, why because, the design space is very huge meaning that, satisfying all these constraints, the there can be a number of designs that satisfy all these constraints and we have to choose that design point which is optimal. Therefore, we need efficient algorithms for doing this and hence again computer aided design.

(Refer Slide Time: 15:34)

## Modeling, Design and Analysis

- **Modeling is the process of** gaining a deeper understanding of a system through imitation. Models express **what a system does** or should do.
- **Design is the structured creation of artifacts.** It specifies **how a system does what it does.**
- **Analysis is the process of gaining a deeper** understanding of a system through dissection. It specifies **why a system does what it does** (or fails to do what a model says it should do).

The three phases are overlapped and iterative

Embedded Systems 6

Now, therefore, how do such automation proceed in a step by step fashion such automation proceeds through 3 distinct steps. So, computer aided design typically in an embedded system typically proceeds through 3 distinct steps modelling design and analysis. So, what is modelling? Modelling is the process of gaining a deeper understanding of a system through imitation. So, we do not build the entire system, but somehow we imitate the working the functionality of the system to understand what it is output should be sometimes, when such models are very nicely and formally specified it allows according by construction synthesis procedure to and implementation as well.

However, modelling again essentially is the process of understanding the system. So, it makes an abstract system, we do not build the actual system the properties that we think are important we model the system with those properties and check whether the, if the design can satisfy those properties are not. So, models express what a system does or should do. So, it describes the functionality of the system in a systematic manner.

Now, after modelling comes design, design is a structured creation of artifacts. So, in the modelling phase, we have described the functionality or the tasks that the system should do and what is the sequence of tasks that the system should do. Now, all these tasks must be suppose executed on a computing platform. It will need a processor, it will need to store data, it needs input data and it needs to store output data. Therefore, it requires



memories. It needs to communicate with other components and with the external world and hence it needs communication devices such as buses.

So, all these are artifacts, which need to be chosen depending on what is the functionality that we need to implement. Therefore, after the functionality is defined and we have chosen the artifacts to implement those functionalities, we get a design and we also specify in the design what will be the sequence of activities say, of let us say many tasks that we are we have implemented on a particular artifact say a processor ok. So, this encompasses the design therefore, the design specifies how a system does, what it should do.

Finally, after the design, we have come up to a first cut design. Let us say, after modelling and we have come up with the first cut, cut design of what should be the artifacts that should implement those design. We need to analyse whether the design meets. Say, the performance constraints the other non-functional constraints, say constraints on energy consumption etcetera.

So, that is done through a process called analysis. Analysis is the process of gaining a deeper understanding of a system through dissection ok. Through dissection and finding out at each step why a system is performing in a certain way. So, therefore, it specifies why a system does, what it does or fails to do what the model says it should do.

Now, if it finds out, when it finds out why a system let us say fails to do a certain thing, it may be for example, because a function cannot be displayed or cannot be implemented on a certain type of processing platform. Let us say and we need to change the processing platform; that is what our analysis says.

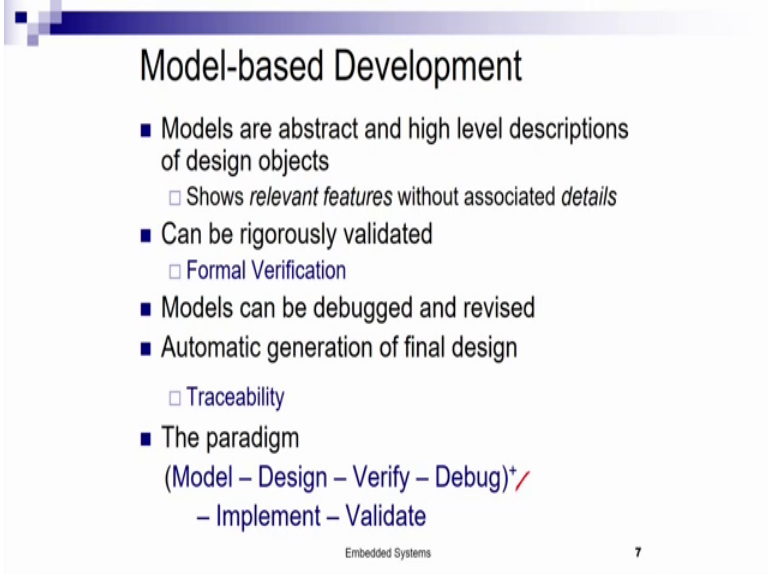
So, therefore, we need to go back to the design and change the processing platform and then come back and do the analysis again or it may find out that the requirements that we had specified are incorrect and no matter what kind of, design artifacts you use. Those requirements that we had specified in the model itself cannot be used to do the design.

So, therefore, we need to correct the mod a correct at the modelling phase itself. So, analysis now tells me that whatever requirements, you had told in the modelling phase was not correct. So, the functional model was not correct. Therefore, we need to correct the functional model do a redesign and then do the analysis again.

Hence, the 3 phases are overlapped and iterative. So, we go for a modelling to design to analysis we after design, we can come back to modelling change the model go back to design or from analysis, we can go back and do a redesign or a remodelling ok. So, these are the different design phases.

Now, we come to model based development, the first phase of design that we told modelling we come to model based development the models are abstract and high level descriptions of design objects.

(Refer Slide Time: 20:42)



**Model-based Development**

- Models are abstract and high level descriptions of design objects
  - Shows *relevant features* without associated *details*
- Can be rigorously validated
  - Formal Verification
- Models can be debugged and revised
- Automatic generation of final design
  - Traceability
- The paradigm  
(Model – Design – Verify – Debug)<sup>+</sup>  
– Implement – Validate

Embedded Systems 7

It shows relevant features without the associated details just to take an analogy. For example, let us say your Google maps, that you use every day is a model of a certain geographical area. It does not take into details. Let us say that terrain, may not take let us say that terrain, the exact water bodies many details of the actual geographical region, it will just not take into account. However, it will take into account what is important for it. For example, the road network the distance between 2 places etcetera. So, this is a model the, your Google maps is a model of a region for a certain purpose.

For example, to find out what is the distance between 2 places and that it adequately and nicely does hence models are abstract and high level descriptions of design objects this is same for example, let us an embedded system the example I took during the during the intro video that of a car parking system, that in a car parking system, you need an embedded system that keeps into account the number of cars at any given time and we

build a finite state machine to which in; which the states are the number of cars at any given time and the transition represent a car going out or a new car coming in it does not model the actual car parking system.

However, it has a machine which it is a finite state machine model, which has relevant details relevant features to model the computing embedded system that we want to design a program or an embedded computing system that keeps into account at any point in time the number of cars. Therefore, models show relevant features without associated details.

However, models if nicely stated can be rigorously validated. So, if I have a formal let us say a state machine design, then I can formally verify whether all properties are verified whether all properties are verified can there be a deadlock with cars coming and coming in and going out in the car parking system. So, I can verify certain properties like that can an output of a data. We all will can always be received within say a certain time say 100 milliseconds.

So, these properties may be validated, if we have a formal modelling of the system. Therefore, models can also be debugged and revised. It allows sometimes it allows automatic generation of the final design, we will see 1 or 2 lectures later how modelling can allow an automatic correct by construction design when nicely and formally specified well take an example of that.

Hence, model based development. What is the paradigm? You first model and then you design you verify the correctness of what you have designed and then you debug and then you may need to go back to the model other design and this may continue one or more times and hence, the plus and after you have design. Hence, the plus over here and after you have ensured that the design is correct you implement it on actually implemented on a platform manufacture it and then you validate the performance of that is of the manufactured or implemented system the prototype system or the manufactured system.

So now, we come to a bit more details on FSM based behavioural modelling.

(Refer Slide Time: 24:40)

## Embedded Systems

### FSM Based Behavioral Modeling

- Behavioral or Functional Model: Control flow among components
- Architectural or Structural Model: Interconnection among components
- Finite State Machines – One of the most popular behavioral modeling formalisms
- The system moves in discrete steps from one state to another by executing transitions
  - **State:** denotes a specific valuation of the system variables
  - **Transition:** denotes a change in valuation

```
graph LR; Off((Off)) -- "press?" --> Light((Light)); Light -- "Press?" --> Off;
```

Torch Light Control

Embedded Systems

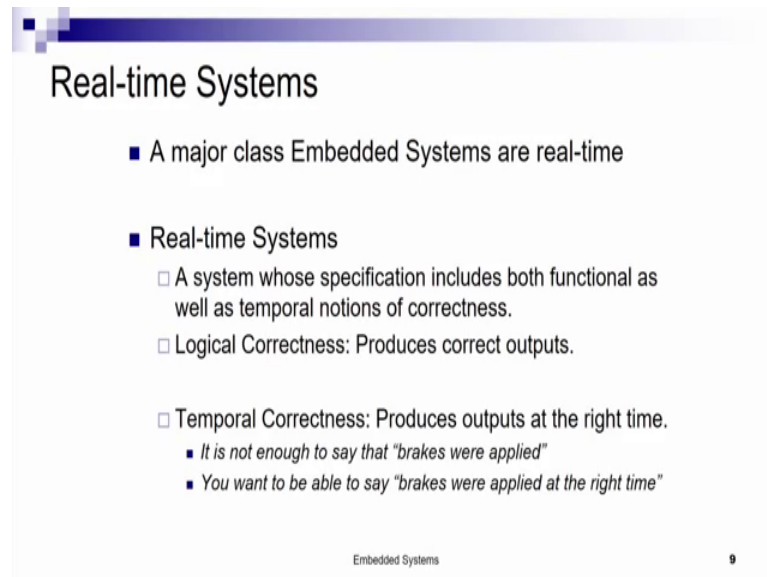
So, what does behavioural modelling behavioural or functional model? It is a control flow among components. It tells what are the distinct states, that an embedded system can be in and how the change of states can occur. So, there are different modelling of the behaviour; one is the functional modelling which it tells what are the functions that it should do. You can also model the architecture and get a structural model. So, it which will tell you the interconnection among components and finite state machine is an very important behavioural modelling formalism which we will study in this course.

So, in a finite state machine, the system moves from discrete step in discrete steps from one state to another by executing transitions. So, as I told you before as well, the states tell you the static properties or it denotes a specific valuation of system variables and the transitions tell you the dynamic properties. It tells you how the change in states occur over time.

For example, let us say that, we want to design a very simple torch light controller a small embedded computing device that will be put inside your torch light. Firstly, the switch is off and then you when you press the switch you get to the lighted state. So, both off is a static property is a static state of the device, and then when you press the button then it takes you to another static state, which is light that the torch is now glowing and then you press. When it when you press it again, it again takes you back to another static state which is off state, back to the off state.

So, this is how you this is this is a very simple example. But, it tells you that what are the; what is the essential idea behind modelling of the functionality of a system using state machines.

(Refer Slide Time: 26:50)



**Real-time Systems**

- A major class Embedded Systems are real-time
- Real-time Systems
  - A system whose specification includes both functional as well as temporal notions of correctness.
  - Logical Correctness: Produces correct outputs.
  - Temporal Correctness: Produces outputs at the right time.
    - *It is not enough to say that "brakes were applied"*
    - *You want to be able to say "brakes were applied at the right time"*

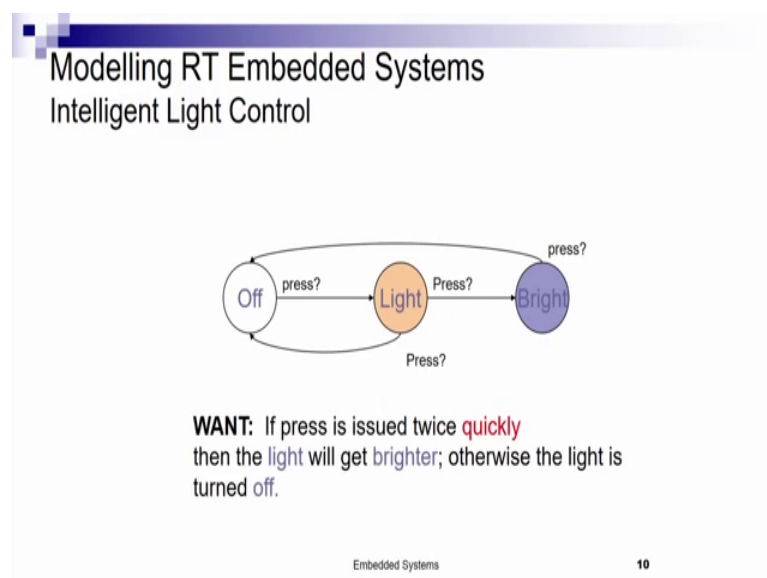
Embedded Systems 9

A major class of embedded systems are real-time in nature. So, what are real-time systems? A systems whose specification includes 2 notions of correctness a fond of it has to be functionally correct; that means, it should provide correct outputs and it has to produce those outputs within time. So, a system whose specification includes both functional as well as temporal notions of correctness logical correctness, it produces correct outputs and temporal correctness produces outputs at the right time, in a real time systems are characterized by things like this.

It is not enough to say that breaks in a car were applied, you want to be able to say that brakes were applied at the right time to make another more say this pertinent example, let us say you want to design the anti-lock braking system in a car. So, the antilock braking system what it does is that, it allows the car to stop within a short distance without the car; car locking it is wheels and skidding. So, it avoids the skidding of the car which happens due to the locking of the wheels. When you mechanically press the brake very hard, how does it do? It releases the wheels within short spans of time after the brake is pressed.

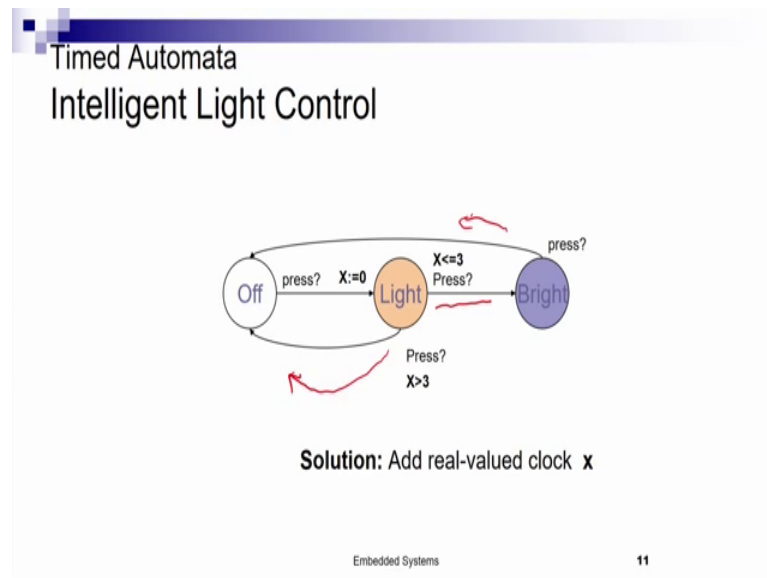
Now, let us say you have a design constraint like this. So, once the brake is pressed, after the brake is pressed, you need to release the brakes within let us say the next 100 milliseconds to stop the wheels from locking. Now, therefore, you have to complete the functionality you have to do the computational job of releasing the brake from locking and you need to do it within the next 100 milliseconds. So, this is a real time requirement where you have a functional requirement that you have to complete a job and produce an output, which is releasing of the brake and you have to do it within a certain deadline which is 100 milliseconds in our case.

(Refer Slide Time: 29:08)



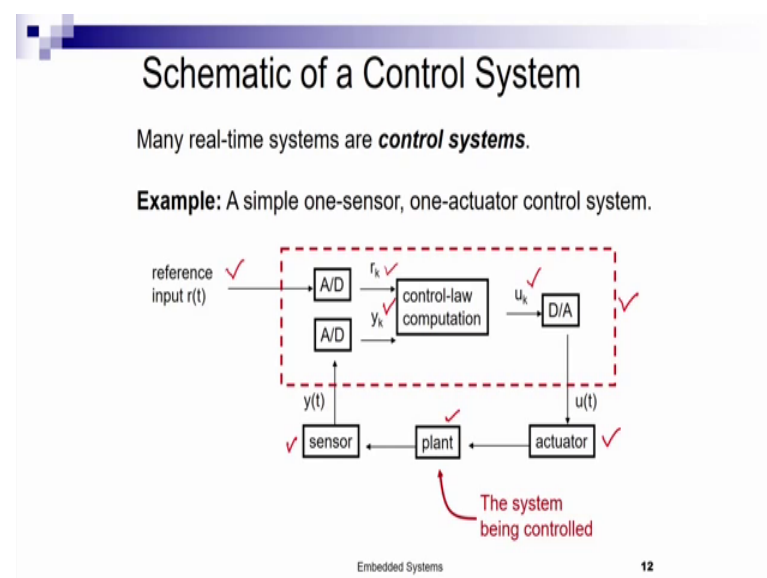
Now, how do you model a real time embedded system? For example, let us take the torch light. Example, a bit further our simple torch light now has 3 states one is the off state, the one is the next is the lighted state and the further you have a brightly lighted state. Now, what do you want? If the press is issued twice quickly, then the light will get brighter. Otherwise, the light will be turned off the question is how quickly this model does not specify anything. Now, how can you do it? You have to change the finite automata to a timed automata, which allows the specification of real valued clock variables.

(Refer Slide Time: 29:42)



Now, this one says that suppose you are at time 0, when  $x$  is equal to 0 and you are in the off state and you press the switch at time 0, you go to the lighted state and then if you press the switch. Again within the next 3 time units you go to the bright state if you do not press within the next 3 time units you go back to the off state. So, here if you press it within the next 3 time units, you go to the bright state. If you do not press it within the next 3 time units, you go back to the off state and in any case, at the bright state, you press the switch again, you go back to the off state ok. This is how you specify a model or real time embedded system.

(Refer Slide Time: 30:44)



Now, we come to another important aspect many real time embedded systems are control systems that it controls because it many real time in many embedded systems control a bigger physical system right. So, these are control systems. For example, let us consider a simple one sensor one actuator control system. So, here you have a sensor here you have an actuator here you have a plant the physical system and here this red dotted box you have the controller inside that. Let us say this is a simple cruise controller in your car and the objective of the cruise controller is to maintain the speed of the car at say 40 kilometres per hour.

So, my reference input here is 40 kilometres per hour at which I want to keep the speed the this reference is input is analog to digital converter and  $r_k$  is the input to the controller computation and let us say this plant is our car and this one the sensor is our wheel speed sensor. And therefore,  $y_t$  tells me what is the current speed of the car and that is ad converted and we get  $y_k$ , which is the current speed of the car and then the control of computation tells me what.

Firstly, it checks what is the error. That means, this was the desired speed and this is the actual current speed and therefore, what is the difference in speed and let us say after that this  $U_k$  tells me how do you change the throttle of the car so, that the speed of the car can reach the desired speed.

So, this  $U_k$  is the digital output corresponding to the thing that, you want to say what is the throttle that is needed to change the speed to the desired speed and that is again digital to analog converter and this output goes to the actuator which is the plant. Again the car and the throttle is changed and again we go on doing and go on checking what is the current speed what is the desired speed, what is the error and what should be the throttle and this goes on in the control system and this controller, if it correctly functions will be able to effectively control and maintain the speed at around 40 kilometres per hour.



(Refer Slide Time: 33:18)

**Example of a Real-Time Task**

**Pseudo-code for the control task:**

```
set timer to interrupt periodically with period T;  
at each timer interrupt do  
  do analog-to-digital conversion to get y;  
  compute control output u;  
  output u and do digital-to-analog conversion;  
end do
```

*T* is called the **sampling period**. *T* is a key design choice.  
Typical range for *T*: seconds to milliseconds.

Embedded Systems 13

Now, let us now take a deeper look into the control law computation or the control task that we had remember that in the model before the control computation is a functionality and it will be modelled in terms of a finite state machine and that will be converted into a code sequential that is also a sequential model of a formal sequential model, which is in the form of a program or code and that code will finally, be at will be referred to as a task or control task and will run on a processor.

Let us take a look at this control task. So, what is happening in this control or computation or this control task, we set the timer to interrupt periodically with a period  $T$ . So, every  $T$  in units of time, I check what is the current speed of the car. Say at each time what interrupt we do analog to digital conversion of  $y$  to know in  $T$  in the digitized version of the current speed, then we compute the control output.

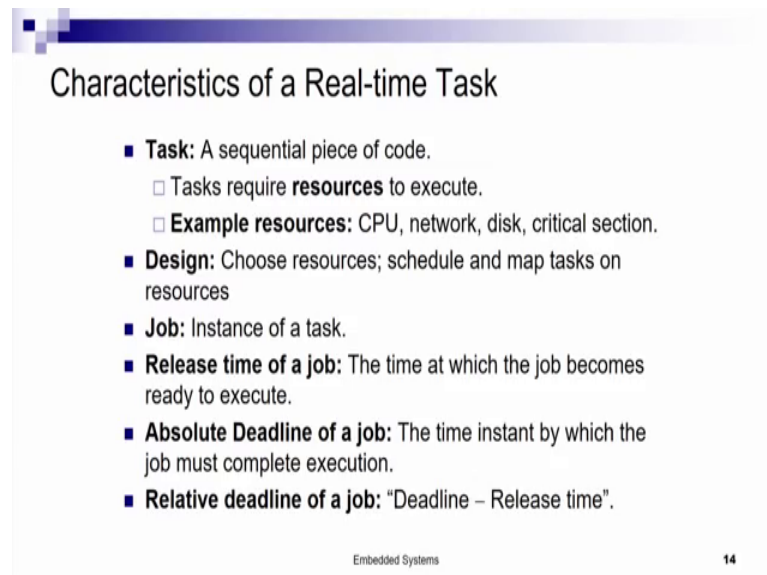
So, that the current throttle needs to be computed and output, the output  $u$  is then digital to analog conversion converter and we feed it to the actuator this is what is happening again and again every time period  $t$ . Therefore,  $t$  is called the sampling period. After every after every sample period interval, you are checking certain parameters of the physical system. For example, here the speed of the car  $T$  is a key design issue.

Why is it is? It is a key design choice because, suppose you check the speed of the car at long intervals of time, then the speed of the car can from it is desired speed by a large

amount, which you do not want on the other side let us say you check the speed of the car extremely frequently.

And therefore, the computation has to be done very frequently which will require a very powerful computer or a platform on which to do the computation which may be unrequired which may be not required for the system that you have and because it is sufficient to let us say check the speed of the car every mm say 500 milliseconds or so or 1 second. You do not need to do it every let us say 1 millisecond hence the value of T is a key design choice typically the range for T could be between 10 seconds to milliseconds.

(Refer Slide Time: 36:04)



**Characteristics of a Real-time Task**

- **Task:** A sequential piece of code.
  - Tasks require **resources** to execute.
  - **Example resources:** CPU, network, disk, critical section.
- **Design:** Choose resources; schedule and map tasks on resources
- **Job:** Instance of a task.
- **Release time of a job:** The time at which the job becomes ready to execute.
- **Absolute Deadline of a job:** The time instant by which the job must complete execution.
- **Relative deadline of a job:** "Deadline – Release time".

Embedded Systems 14

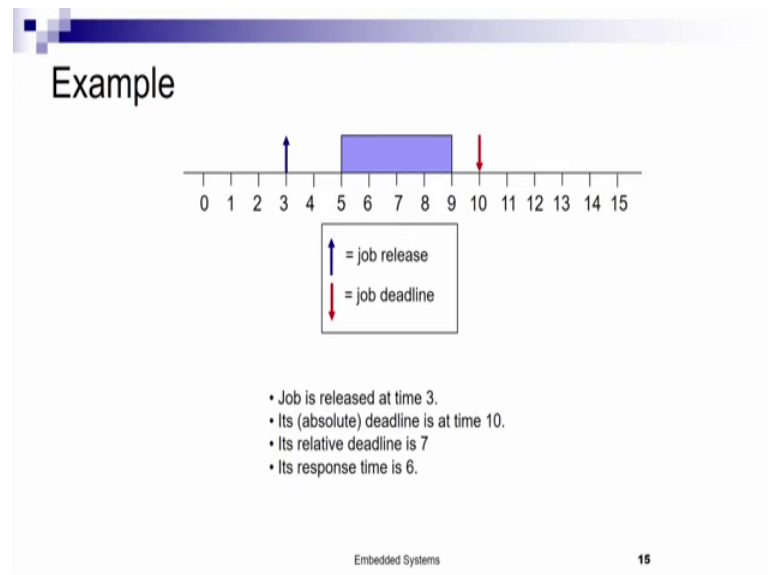
Now, with this understanding of the control task we go deeper into the issues of few issues related to control tasks to a real-time control task. So, task as we understood is a sequential piece of code tasks require resources to execute, it will require memory, it will require CPU, it will require non-critical section or non-pre-emptible resources it will require network to communicate with others. So, therefore, tasks will require resources to execute it is a piece of code which will require resources to execute.

So, in the design phase after we have we have found out the tasks in the design phase we choose we choose resources for example, appropriate CPU's appropriate network protocol appropriate disk or memory etcetera. We choose and then schedule and map the tasks on to these resources because multiple tasks may use these resources. Therefore, we have to schedule and map the tasks on the resources and we said that the task is this

piece of code and this task is executed repeatedly every  $T$  time period. So,  $T$  is the sampling period and every  $T$  time units. I am executing this task once. So, job is such an instance one instance of the execution of the code of a task.

So, job is an instance of a task release time of a job the time at which the job becomes ready to execute is called the release time of the job, absolute deadline of a job, the time instant by which the job must complete execution is called the absolute deadline of the job. And we also have a relative deadline of the job which is the deadline relative to its start time all these parameters in the form of an example.

(Refer Slide Time: 38:14)



Let us say you have a job the job is released at time 3 the start of the job happens the actual start of execution of the job happens at time 5 and the completion of the job happens at time 9 and the deadline before which the stipulated deadlines before which the task should complete is 10. So, the job is released at time 3 its absolute deadline is 10 its relative deadline is 7. Why it is 10 minus 3? It is the deadline relative to its start time.

So, its deadline is 7 and its response time is 6. So, after the release of the job when does the output of the job visible output from the execution of the job visible at time 9, the output is visible and hence the response time is 6.

(Refer Slide Time: 39:11)

### Real-Time Periodic Task

- Task : A sequence of similar jobs
  - Periodic task  $(p, e)$ 
    - Jobs repeat regularly
    - Period  $p$  = inter-release time ( $0 < p$ )
    - Execution time  $e$  (maximum execution time;  $0 < e < p$ )
    - Utilization  $U = e/p$

Embedded Systems 16

Now, we come to the definition of a real time periodic task. These tasks, these repetitive tasks where the jobs come again and again could be of different types. It could be periodic a periodics sporadic different types, we will look at them, but currently we take just a look at periodic tasks, which we saw just now that at exact fixed intervals at intervals inter arrival job inter arrival time of the jobs are fixed and constant.

So, one instance of the job comes every  $p$  time units here a periodic task whose execution time is  $e$  and the that the task or job comes the job comes every  $p$  time units the jobs repeat regularly period  $p$  is the inter release time,  $p$  has to be greater than 0. Obviously, execution time  $e$  is the maximum time that the job can take you.

Remember that, before the execution of the job we do not know exactly how much time the execution will take. Because, it will depend on the resource the task is executing the actual time will vary on the data as well. The input data to the program because it can have different control parts to the through the program so, the actual time is not exactly known. However, we can have an estimate of the time that will that it will take in the worst case called the worst case execution time and it has to be something that is between  $e$  and  $p$ .

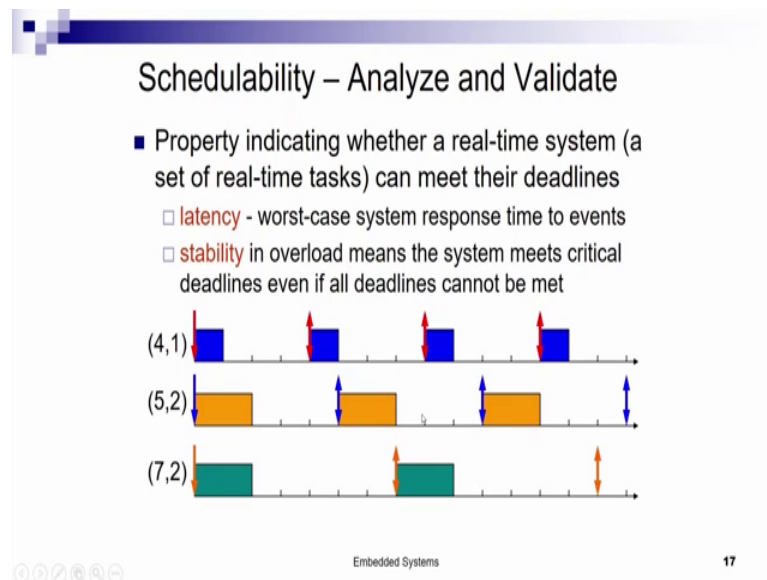
The execution time if it is more than  $p$ , it will never be able to complete before the next job arrives. Hence, execution time the worst case execution time has to be between 0 and  $p$  and then the utilization  $u$  is equal to  $e$  by  $p$  utilization  $u$  is a measure of what part of the

capacity of a processor that that I need to allocate for executing this job. For example, if  $e$  is equal to  $p$  then  $U$  becomes 1, which means that one processor has to be dedicatedly allocated for the execution of the job. If  $U$  is greater than 1, then  $e$  becomes greater than  $p$  and the job can never satisfactorily then the jobs or the tasks can never satisfactorily be executed on this processor.

So, here is an example of a task in with where we have shown 3 jobs. So, this periodic tasks arrives every 5 time units and it has to be executed after it is arrival within 5 time units. So, the first job executes at between 0 to 1 the execution time is obviously 1 in this case the second job executes between 6 and 7, and the third job executed between 11 and 12 in this example.

Multiple tasks can execute on the same resource as we said. So, a resource can be shared among multiple tasks and when a multi when multiple task share a single resource, we have contention for the shared resource who will execute when who will execute earlier who will execute later that has to be decided. We need to have we need to decide a sequence of execution for the jobs of different tasks on this shared resource.

(Refer Slide Time: 42:45)



So, how do we understand whether a particular task or a set of jobs all jobs that are that I have that I am executing on a resource will always meet, it is required constraints in the form of say deadlines and stability. So, that is analysed through a mechanism called

schedulability. So, schedulability what is it? It is a property which indicates whether a real time system or a set consisting of a set of real time tasks can meet their deadlines.

So, deadlines in the form of latency that is worst case system response time to events; that means, after let us say the event an event can be the arrival of a job. So, after the arrival of a job can I say that given these set of tasks, I will always be able to meet the deadlines requirement.

So, given that a task arrives every 4 time units. Say, if it is arriving at 0 can I always say that it will always be able to finish before 4, when it arrives at 4 can I say that it will always be able to finish before 8 when I have 1 task in this case. For example, I possibly will be able to say that yes I have just one thing to do only one task and I have 4 units of time and I will be able to always meet it.

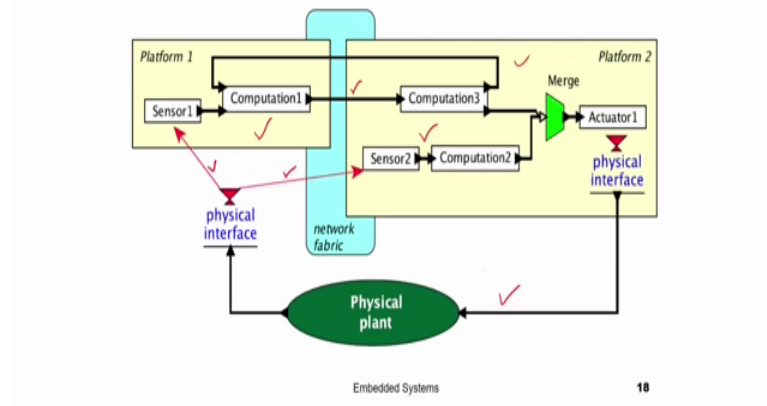
But, how do I say when I say I have 2 tasks; one task it has execution time one comes every 4 time units and another task execution time of 2 comes every 5 units. What is the mathematics that will allow me to say that, yes these 2 tasks together all jobs of these 2 tasks together will always be to meet all deadlines on a given resource.

What happens when I have 3 tasks now? So, we have to study how schedulability analysis is done. So, in the face of and be able to say what will be the maximum latency and the latency constraint will be met and whether with the system will be stable whether this schedule will be stable. That means, during overload stability during overload and means that the system will meet will be able to meet critical deadlines even if all deadlines are not met.

So, stability tells me that when I have a overload and what is the meaning of overload; that means, that the task that I have a many tasks have arrived and I cannot meet all deadlines the total utilization has become more than the capacity. Do I my does my schedule meet the stability criterion? Meaning that, can I still meet deadlines of critical tasks if even if I am not able to meet all that is an analysis which is done through stability, it is called stability analysis and we will look at it.

(Refer Slide Time: 45:54)

## Schematic of a Cyber-Physical System



Now, I will digress from the definition of from embedded systems to what are called cyber-physical systems and IoTs I am digressing here because, in the subsequent lectures. We will no more going to speak separately about cyber physical systems or IoTs. But these are these discussions are relevant to embedded systems design because they are so they cannot be they can be called certain generalizations of embedded systems itself ok. So, or specializations of embedded systems itself, what is a cyber-physical system an embedded system?

If we look at it is a computation and a physical system, I have a physical system and a computing platform controlling that physical system that is an embedded system. Now, a cyber-physical system is a cyber-component controlling a physical plant and how is the cyber component different from the computation component of an embedded system. Now, the cyber component not only includes computation, but also network or communication between components. So, there can be multiple computing platforms which communicate between through a network among themselves.

For example, you may have multiple components each having it is own computation capability within a car and they may collaborate with each other and produce results for a third component.

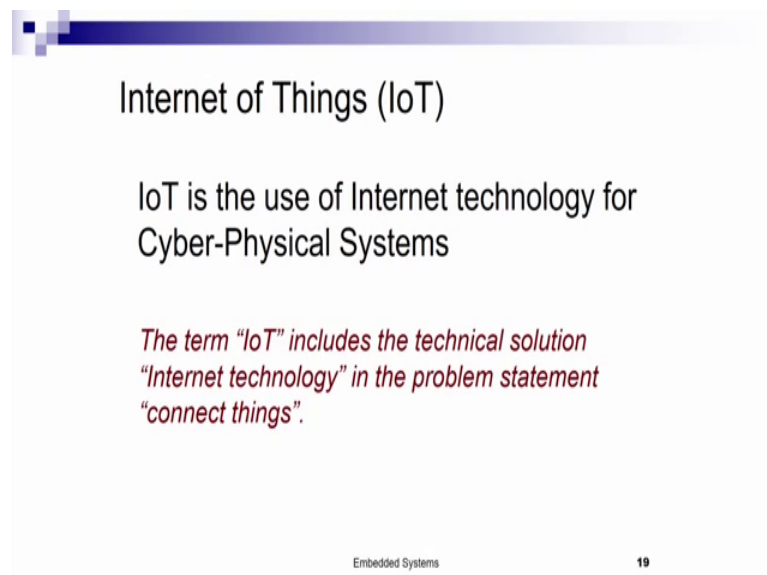
For example, let us say that I have this physical plant car and like in the previous example I have sensors which take physical parameters from the car; one is let us say the current wheel speed and the other is, let us say the current air fuel mixture in the fuel

injection system. So, this is the this sensor computes the air fuel current air fuel mixture in the fuel injection system and this one computes the our previous as our previous example, this one computes the current speed of the car and generates the throttle that it needs to generate to go to a desired speed.

Now, the computation 3 here then let us say compute computation 3 and computation 2 together then let us say tells me what should be the change in air fuel mixture in the fuel injection system so, that the desired throttle can be achieved. So, therefore, this cruise control system and the fuel injection system they collaborate with each other to determine what should be the final the talk with each other over a network to determine, what should be the final air fuel mixture and that is again the actuator out the actuator output and produces the final required air fuel mixture and that is fed to the physical plant again that is a car.

So now, we have just have we have gone from embedded systems to cyber physical system just to mean that the computation part is not on the computation, but it is cyber which includes a network communication along with computation and it controls the physical component. So, this is a cyber-physical system.

(Refer Slide Time: 49:21)



Internet of Things (IoT)

IoT is the use of Internet technology for Cyber-Physical Systems

*The term "IoT" includes the technical solution "Internet technology" in the problem statement "connect things".*

Embedded Systems 19

And, what is an IoT or an internet of things again IoT can be defined as the use of internet technology for cyber physical systems. Here the term IoT includes the technical solution internet technology in the problem statement connected things. So, these things



are could be defined as individual embedded systems with the computing capability. Now, these things communicate over the internet to do something that something which includes both of them, but is different.

For example, let us say you have a wearable medical embedded system on a person's body which finds out the current mood of a person and you have another embedded system. Let us say, a light which can which can automatically dim itself our torch light example, for example, which can automatically dim itself or brighten itself.

Now, let us say both these things are connected to the internet and we have a third system comprising which includes both these systems that is the wearable device and this hour programmable light. These 2 systems that I have it together and depending on the mood of the system what does this third system, which includes these 2 system does this third system is essentially the IoT system.

So, what is this third system doing? It finds out the mood current mood of a person and brightens or dims the light, it is an arbitrary example, but it tells you that it is the IoT is the system that includes these things and provides a solution using internet technology.

So, IoT is the use of internet technology for cyber-physical systems and again if we come back to this in the term IoT. Therefore, includes a technical solution the technical solution is that the system will be able to automatically dim or brighten the light will be able to automatically dim or brighten depending on the input provided by the variable device that tells me the current mood of the person.

So, the term IoT includes a technical solution internet technology in the problem statement connected things. So, why is such connectivity important?

(Refer Slide Time: 51:54)



## Why is Connectivity Important?

- Manufacturers can add value to their products
  - Customers to remotely connect to, control, and monitor their products – say through Apps
  - Provide APIs to allow third parties to develop additional connections to those products.
    - Expands a product's value by offering not only new functionality but also new ways to interact with it
    - Systems of Systems
- Make easy product updates
  - Upgrades will increasingly be delivered via software
    - Example – Over-the-air updates for Model S of the electric-car company Tesla - Autopilot update.

Embedded Systems

20

In embedded systems, in the products related to embedded systems, because of various reasons and we jot down a few of them because, IoTs are going to come in a big way and all embedded systems designs may have to consider the component of connectivity and IoT in their designs. So, connectivity is important why manufacturers can add value to their products through connectivity.

Let us say, it allows customers to remotely connect to and control and monitor their products say through apps.

For example, let us say I have an app which can control the AC in my home even remotely. So, I am in a car 10, 10 minutes away from my home and I say through an app that I have to on the AC. So, that when I actually reach home I have a cool room.

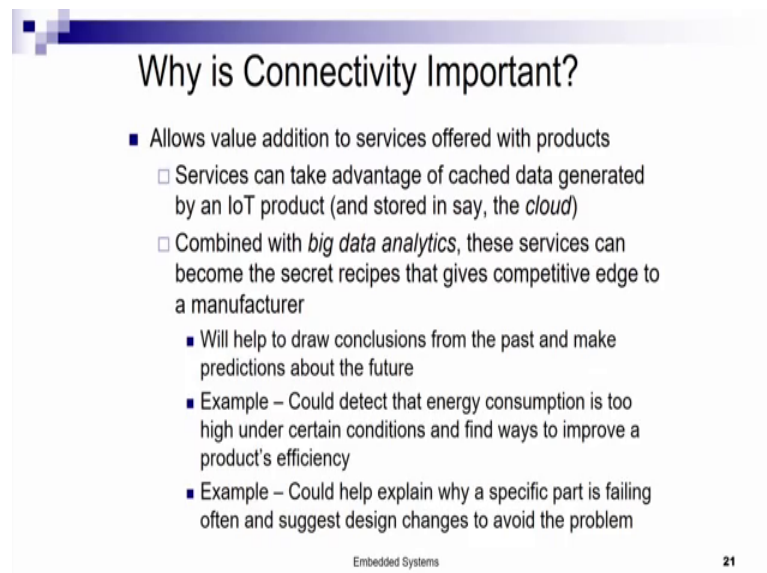
So, customers can remotely connect to control and monitor their products through say apps it provides APIs to allow third parties to develop additional connections to their products. So, if you have big systems which has which includes many systems say a smart home system and in the in that there are various components in it various small small systems and third parties providing these components, then third parties can automatically provide updates on these components and provide value-added services. So, it can provide APIs to allow third parties to develop additional connections to these products ok.

So, I have a system of systems and it allows APIs and these APIs I can have additional connections to these products, why will this product it expands a product value by offering not only new functionality but also new ways of interacting with it. Suppose, I can allow an API to update my system so, third parties can update my system and enhance it is functionality this is what I come to in the next. So, it allows easy product updates.

Upgrades will increasingly deliver be delivered via software for example, over there updates for model S of them of the electric car company Tesla Autopilot. So, the autopilot update is a software the autopilot update is a software which can be provided over the air. So, over internet in the at night, let us say the car can be the autopilot can be updated autopilot of my car it is sitting in the garage, but over the air over the internet my autopilot software of my car can be updated.

So, all my systems nowadays are softwares my suspension the correctness of suspension everything are softwares and they can be automatically updated over the end, if I allow IoTs or connection through the internet.

(Refer Slide Time: 54:59)



**Why is Connectivity Important?**

- Allows value addition to services offered with products
  - Services can take advantage of cached data generated by an IoT product (and stored in say, the *cloud*)
  - Combined with *big data analytics*, these services can become the secret recipes that gives competitive edge to a manufacturer
    - Will help to draw conclusions from the past and make predictions about the future
    - Example – Could detect that energy consumption is too high under certain conditions and find ways to improve a product's efficiency
    - Example – Could help explain why a specific part is failing often and suggest design changes to avoid the problem

Embedded Systems 21

It will allow, additions addition to allows value addition to services offered with product services can take advantage of cached data generated by an IoT product and stored in say the cloud. For example, let us say you have a steel plant and that steel plant is an IoT because various portions. It is an industrial IoT where various embedded computers are

coordinating among each other and taking the design from raw material to manufactured steel ok.

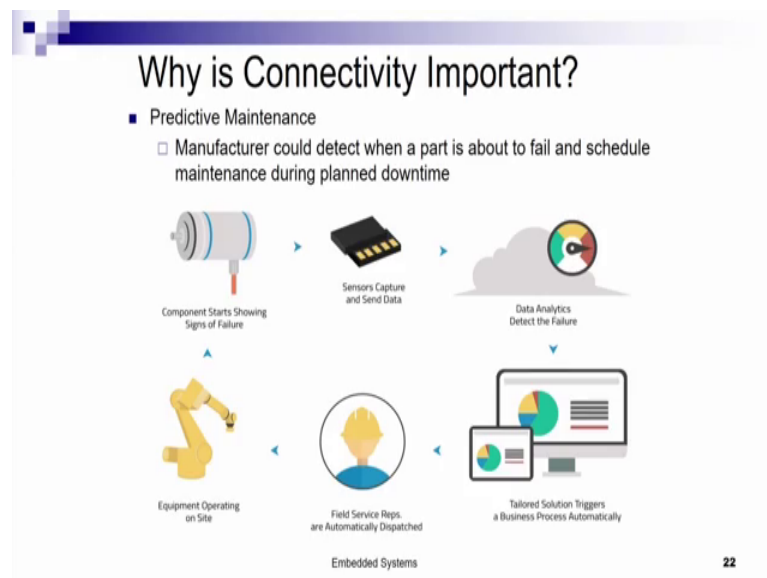
Let us say, now the data corresponding to various parameters at different stages of development of the steel is taken and put into the cloud for better analytics.

For example, combined with data analytics these services can become secret recipes that gives competitive edge to the manufacture. It will help to draw conclusions from the past and make predictions about future with respect to the steel plant. Let us say, it can say that why certain design processes at certain times consume higher amount of energy, it could detect that energy consumption is too high under certain conditions during steel manufacture and find ways to improve our products efficiency.

Now, therefore, this because it is connected through the internet to the cloud it allows better data analytics and hence efficient ways of correcting the behaviour of the system the performance of the system.

It could help explain why a certain part is failing often and suggests design changes to avoid the problem how can this why can this happen because, my data from the product is continuously being taken to the cloud and I am being able to analyse. So, due to such analytics in the cloud, I am being able to explain why a certain part. Let us say, is failing and we can suggest changes to avoid the problem.

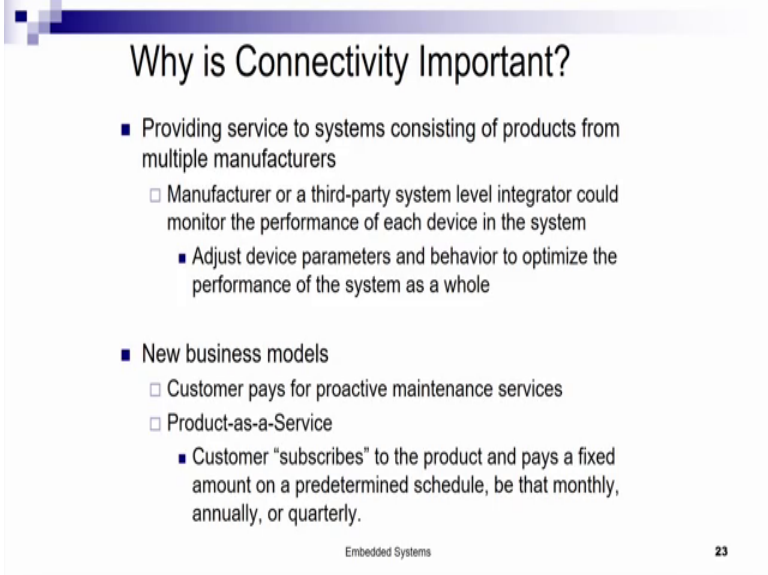
(Refer Slide Time: 57:09)



It can also allow predictive connectivity can also allow predictive maintenance the manufacturer could detect when a part is about to fail. Before it actually fails and scheduled maintenance during a planned downtime. For example, let us say a certain component in an aircraft starts showing signs of failure. It does not actually fail during the flight and then over the internet over connectivity, it can tell to say it is a Boeing aircraft. It can tell the Boeing of service team that a certain component is has started failing as showing signs of failure during flight.

So, how does that happen it the sensors in the sensors in the aircraft capture and send the data and then the data analytics detect the failure the tailored solution triggers, a breathless process automatically the tailored solution tells the service team that there is a failure. And then, the field representatives are ready and the servicing is done whenever let us say, the plane lands on a certain airport before actually the failure is has happened the servicing has been done at the airport. Once the aircraft has landed, this is why connectivity, this is how connectivity has helped in here, it can allow predictive maintenance.

(Refer Slide Time: 58:34)



**Why is Connectivity Important?**

- Providing service to systems consisting of products from multiple manufacturers
  - Manufacturer or a third-party system level integrator could monitor the performance of each device in the system
    - Adjust device parameters and behavior to optimize the performance of the system as a whole
- New business models
  - Customer pays for proactive maintenance services
  - Product-as-a-Service
    - Customer "subscribes" to the product and pays a fixed amount on a predetermined schedule, be that monthly, annually, or quarterly.

Embedded Systems 23

Providing service to systems consisting of products from multiple manufacturers, this we already saw that a system can be may be made up of many other smaller systems. And hence a manufacturer or a third party system level integrator could monitor the

performance of each device in the system and it can adjust device parameters and behaviour to optimize the performance of the system as a whole.

So, it may allow if you have multiple smaller systems all these smaller systems can be monitored and the parameters adjust so, that the entire system as a whole can perform better.

It can also allow newer business models for example; customer can pay for proactive maintenance services as we told. Let us say, for the Tesla car autopilot if proactive maintenance is done the customer may need to pay some money for that service. It can also allow product as a service. For example, you have an embedded system that you have a physical system. For example, it is a battery charging station of your car, which is about to come for your electric car.

Now, this is attached with an embedded computing device which can then which can schedule the service that is the charger that is the charging service to various cars. Then, this product that is discharging system becomes a service, you do not require a personal charger for your cars battery, but you can take the charger as a service discharging, as a service this product now is available as a service.

So, the customer subscribes to the product instead of actually buying it, he subscribed to the product that is the charging and pays a fixed amount on a predetermined schedule on a specific time at a on a specific time of the day or month or annually or quarterly he can schedule for charging of his of his car's battery from a nearby say charging station.

And, because this charging station is has an embedded system and IoT device, which is connected to the internet it can accept such requests and it can provide a schedule of when a certain car will come for service, that is cars come for charging through the chart through the cars charger sorry to the charging station for getting his battery charged, ok.

With this introduction to embedded systems and cyber physical systems and IoTs, we come back to embedded systems design and what we will do in the rest of the lectures of this designs course.

(Refer Slide Time: 61:48)

**Embedded System Development**

- Functionality decomposition and modeling
- Architecture Selection:
  - Choice of processing elements
    - standard / custom / semi-custom HW
    - Memories, Interfacing, communication
  - Mapping of functionality to HW and SW
- Design of an integrated system
  - Such that end-to-end performance goals are satisfied
- Prototyping, verification and validation

Embedded Systems 24

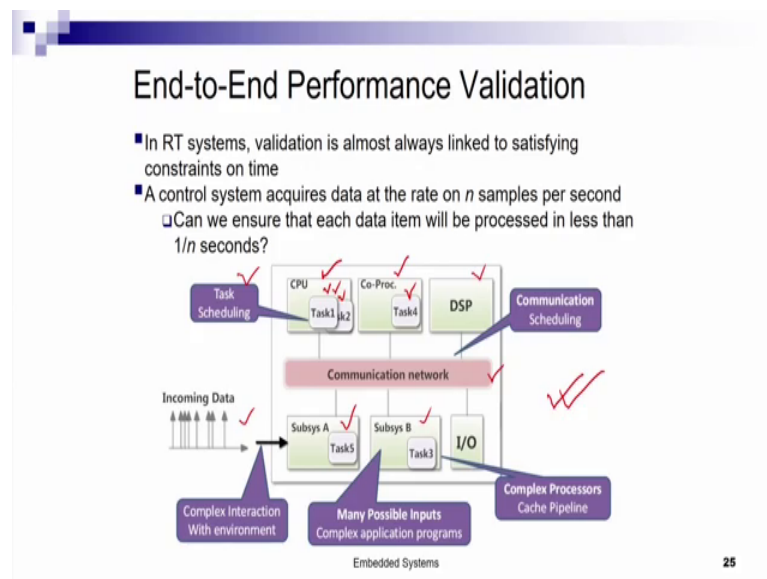
So, the embedded system design phase starts with the specification with the decomposition of the functionality of the system into controllable smaller modules and formally defining the behaviour of each of these modules in the modelling phase. After that, we have an architecture selection phase where the choice of processing elements; that means, we are which type of general purpose processor or a custom made processor or a semi-custom made hardware will be use.

For example, we can use an FPGA we can use a dedicated hardware. We can design an ASIC application specific integrated circuit or we can use a general purpose processor what kind of memories will be used. What kind of interfacing, say Nuart what kind of interfacing we will use what kind of communication buses or interconnection networks will be used. All this has to come in the architecture selection phase.

Then from the architecture selections after the architecture selection is done we have to map the functionality to hardware and software where hardware mapping to hardware will mean that functionality has for that functionality. We need to make a dedicated hardware or an application-specific integrated circuit for mapping to software, say we may mean that we will write c codes for those function unity and executed execute it on a general purpose processor. So, that will be mapping the functionality to hardware and software and then we will do an integrated system level design.

So, we will have an entire system after that, which will include certain say hardware components or certain software components and how they communicate among themselves design of an integrated system such that end to end performance goals can be satisfied. After the design, we have to prototype the system, we have to test verify and validate the system. So, this is an overall glimpse of the design process of an embedded system.

(Refer Slide Time: 64:00)



So, after such a design is done we get a complicated system such as this where let us say we have a general purpose CPU containing 2 tasks task 1 and task 2 we have a dedicated core processor let us say a GPU which has another task. So firstly, we have the tasks which have been modelled. So, after modelling we get these tasks and these tasks are then mapped and scheduled on these processors.

So, the subsystem A and subsystem B are say custom made assets for tasks which need to be implemented in hardware because the performance goals are (Refer Time: 65:01) that let us say cannot be implemented in software on a general purpose processor.

For when multiple tasks are on the same resource in the same CPU, I need to schedule the tasks on the resource because multiple tasks are going to contend for the resource I may require DSPs Digital Signal Processors, I will require a communication medium to talk among different components, I may need to have buffers for incoming data.



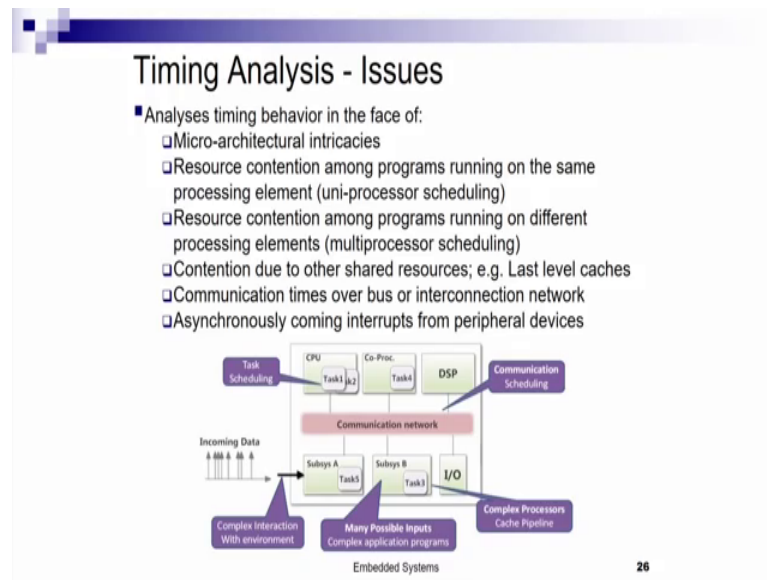
So, it has to it may have complex interactions say periodic aperiodic interactions with the external world. It may have many possible inputs complex app in the application programs can be very complex and hence its needing. For example, here are dedicated hardware it may have complex pipeline structures etcetera ok.

Now, given this total integrated design, we need to verify that validate that end to end performance goals will be satisfied for real-time embedded systems validation mostly not always is almost always linked to satisfying constraints on time. For example, let us say that we have our original cruise control system and the control system requires data acquires data at the rate of  $n$  samples per second.

So, our cruise control system acquires data from the wheel at the rate of  $n$  samples per second then one performance validation goal could be that can we ensure that each data item will always be processed in less than  $\frac{1}{n}$  seconds. Because, every  $\frac{1}{n}$  seconds a new censored input is going to come. And if we do not process it within less than  $\frac{1}{n}$  millisecond the input sense data will be will come at a higher rate than it can be processed.

So, to some to certain extent, I can solve this problem using buffers, but not if not the if this continues continually; that means, the processing cannot be done for a long intervals of time at times less than  $\frac{1}{n}$  seconds or let us. So, we can say that if we can always process within less than  $\frac{1}{n}$  second, then I will not require a buffer at all this design will not require a buffer and all these timing issues needs to be solved in the face of various constraints.

(Refer Slide Time: 67:21)



For example, micro architectural integrates if the processor let us say, has out of order execution then the time that will actually be required will depend on the dependency among the instructions. Say, resource contention among programs running on the same processing element that has to be solved through proper unit processor scheduling resource contention among programs running on different processing elements. It will require multi-processor scheduling contention due to other shared resources.

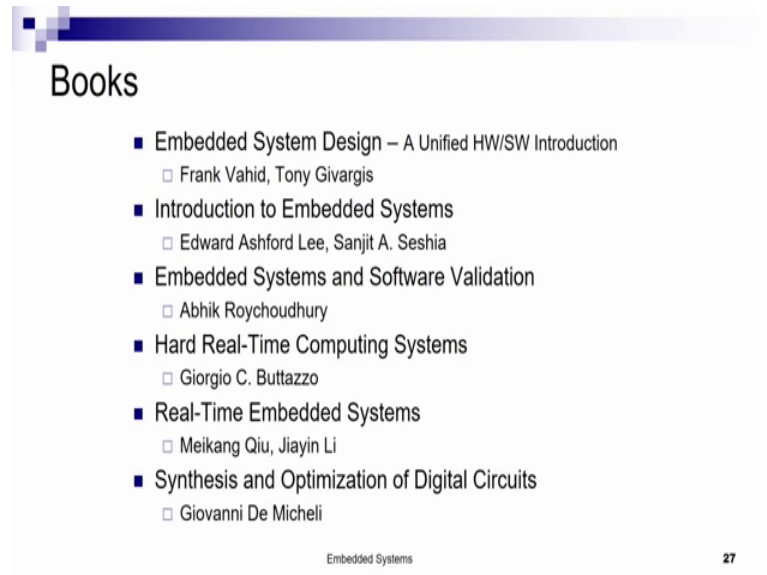
For example, last level caches. So, my instruction was a hit or miss on the cache, it will depending on the hit or miss the time required for an instruction will change. So, the timing issues have to be done the timing analysis has to be done in the face of all these things.

Communication times over the bus and interconnection network what is the time it takes for the messages to pass over the bus. So, how do you schedule it? You need to schedule messages on a shared bus as well and find out in the worst case how much time it will take to transfer a message.

Also, a synchronously coming interrupts from peripheral devices certain interrupts come and stop your processing. So, even in the face of that, what is the bound on the time that certain events will take. So, timing analysis has to take care of all these issues and we will see and we will have an overview over the lectures as to how the design takes care

of all these issues. With this, we come to the end of this lecture. Before ending, will I will just say, I will just tell you the books.

(Refer Slide Time: 69:07)



The slide is titled "Books" and lists seven books with their authors. The books are:

- Embedded System Design – A Unified HW/SW Introduction
  - Frank Vahid, Tony Givargis
- Introduction to Embedded Systems
  - Edward Ashford Lee, Sanjit A. Seshia
- Embedded Systems and Software Validation
  - Abhik Roychoudhury
- Hard Real-Time Computing Systems
  - Giorgio C. Buttazzo
- Real-Time Embedded Systems
  - Meikang Qiu, Jiayin Li
- Synthesis and Optimization of Digital Circuits
  - Giovanni De Micheli

At the bottom of the slide, it says "Embedded Systems" on the left and "27" on the right.

So, these are the books that you may refer for this course along with the lectures; embedded systems design a unified hardware software introduction by Frank Vahid, introduction to embedded systems Lee and Seshia embedded systems and software validation Abhik Roychoudhury, hard real time computing systems Buttazzo, real time embedded systems Qiu and Li synthesis and optimization of digital circuits will majorly require when we talk about hardware design by Giovanni De Micheli. So, with this, we come to the end of this lecture.