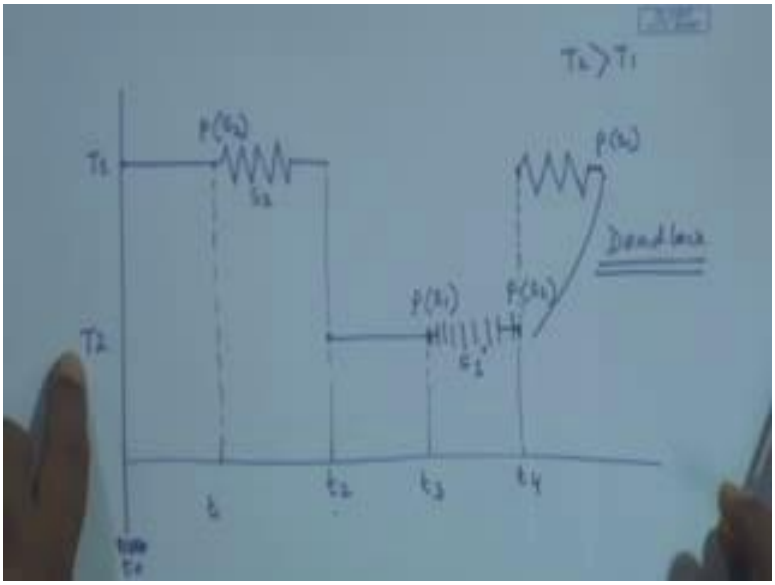
Embedded Systems Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 31
Modeling and Specification – I

Good morning. So, in the last class, we had seen that the priority inheritance protocol can avoid priority inversion. And we had shown that this priority inheritance bit was not set in case of the mars path finder and consequently they were sort of infinite delays, and the system resets were occurring and we are losing data. And that was that was avoided by setting the priority inheritance bit in the mars from the earth station itself and that was avoided.

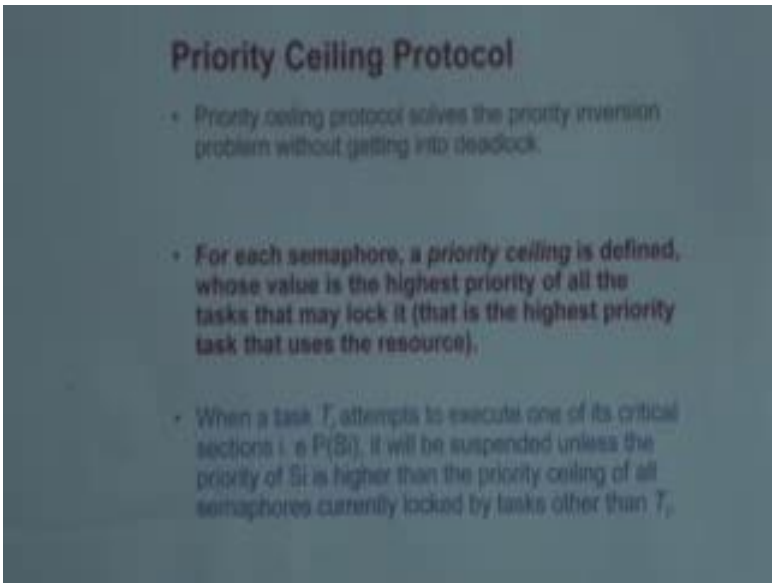
(Refer Slide Time: 01:10)



But nevertheless as we had shown in the last class this scenario that there are two tasks T 1 and T 2 even if and T 2 is having a higher priority than T 1 and there are two critical section which are been accessed by the both processes in a particular sequence. So, first say T 1 started and T 1 entered the critical section S 2, and while it was in the critical section S 2, T 2 the higher priority task started and entered the critical section S 1, no problem up to this, because they are two different critical sections. But, at this point that is a time t 4 the task T 2 which was having higher priority wanted to enter the critical section S 2, which was already acquired by the lower priority task T 1.

So, consequently the priority inversion took place. And T 1 got the priority of T 2. And so T 1 continued with the activity in S 2. And while in S 2 that we asked for the; ask for an access to S 1 which was captured by T 2. Now, you see either way I invert the I change the priority now, there will always be a dead lock situation. So, although priority inversion sorry I am continuously making the mistake, although priority inheritance protocol avoids the priority inversion scenario, but still it is not free from vises that is it deadlock can occur. And in order to avoid that another variety slight modification of the protocol was proposed and that is called priority ceiling protocol.

(Refer Slide Time: 03:16)

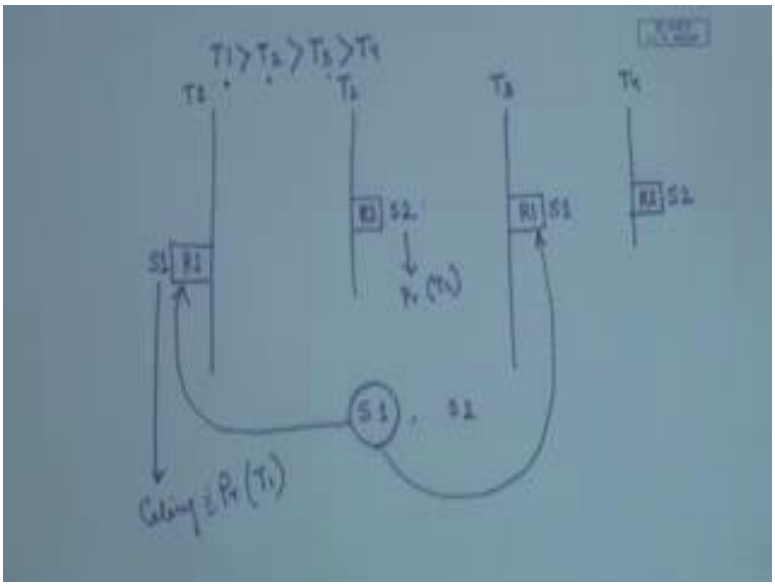


We will rework the earlier example with this priority ceiling protocol, but before that let us try to understand what this priorities ceiling protocol is. The priority ceiling protocols is also priority inversion problem. So, priority inversion is also solved and dead lock is also avoided. What is done in this case, for reach semaphore like in the earlier example, we had two semaphores S 1 and S 2, a priority ceiling is defined. And what is that ceiling, that ceiling is the value the ceiling value is the highest priority of all the tasks that may lock it.

What does it mean, what does this mean, the highest priority of all the tasks that may lock it. You see there are number of tasks all right, a number of task are running concurrently; and each of these task can ask for different resources, and each shared resource is protected by some semaphore. Therefore, at any point of time, when say there

are task T 1, T 2, T 3 out of which T 1 and T 2 are sharing semaphore are sharing particular resource and that resource is being guarded by semaphore say S or S 1. Now, out of these T 1 and T 2, suppose T 1 has the higher priority, therefore semaphore S 1 ceiling value will be that of the priority of T 1.

(Refer Slide Time: 05:23)



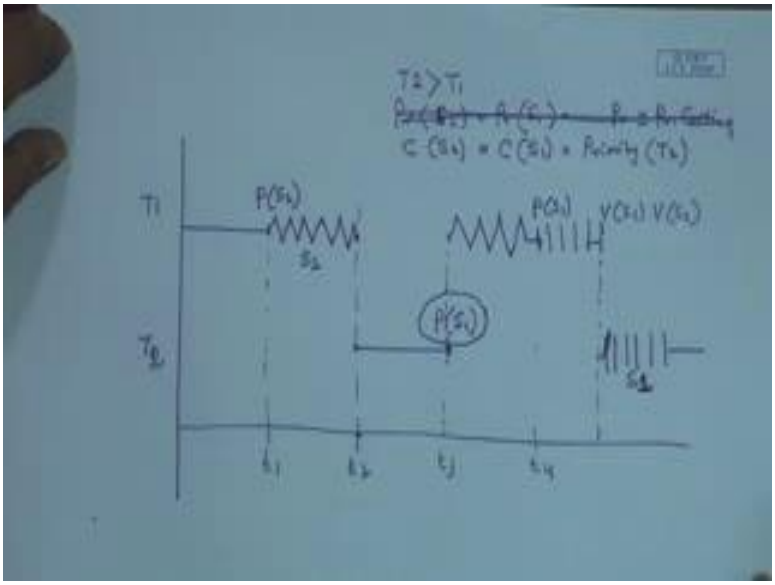
Let me explain once again. So, you see there are number of task T 1, T 2, T 3, T 4 all are executing and there this one requires say resource R 1 which is guarded by semaphore S 1. This one asks for resource R 2 which is guarded by semaphore S 2. This one asks for resource R 1 at some point of time during it execution and therefore, its semaphore is guarded by S 1 and set sorry this resource is guarded by S 1, because that is the shared resource. And I do not know apriori who will request for the resource when. So, T 4 is also asking for resource R 2 which is guarded by semaphore S 2. So, I have got two semaphores S 1 and S 2. What will be the priority of the S 1? Now, suppose out of all this tasks, T 1 has higher priority than T 2; and T 2 has got higher priority than T 3, and T 3 has higher priority than T 4 say.

Now, semaphore S 1 is being held or not actually held right now can be held by task T 1 and task T 3. So, out of task T 1 and T 3, T 1 has got the higher priority. So, the ceiling of priority ceiling of S 1 will be that of the priority of T 1. Similarly, between T 2 and T 4 they are contesting over semaphore S 2, therefore, T 2 having higher priority than T 4, the ceiling of S 2 will be the priority of T 2. Now, let us see what happens. So, once

again let us read this with this explanation. For each semaphore like our S 1 and S 2, a priority ceiling is defined whose values is the highest priority of all the task that may lock that is the highest priority of the task that uses that resources. Clear?

Now, when a task T i attempts to execute one of its critical sections that is say P S I weight on S i will take place p of S i will take place, it will be suspended unless the priority of S i that particular semaphore is higher than the priority ceiling of the semaphores currently locked by tasks other than T i complicated. So, when a task say let me work out this example and we will come back to this. In the mean while, you may like to note down this point where I am trying to match it with my explanation. When a task T i attempts to execute one of its critical section that is tries to execute a P of S i, it will be suspended unless the priority of this particular S i is higher than the priority ceiling of all the semaphores that a currently locked by tasks other than T i. Let us see what happens.

(Refer Slide Time: 09:46)



So, I had tasks T 1 and T 2; and T 2 was having higher priority than T 1. Now, as you can see here, what will happen to the ceiling of S 1 and S 2. Who is using S 1, who is using S 1? T 1 as well as T 2, both of them. And who is using S 2, T 1 and T 2 both. Therefore, according to our definition of our priority ceiling, the priority ceiling of S 1 and S 2 will be both the same and that of equal to that of T 2. We will now do the same earlier task that was shown to you earlier here, the same task, I will be now scheduling and will see

if dead lock occurs or not. Now, as I am showing here, T 2 is having higher priority than T 1, but since both of them are using S 1 and S 2, the priority of S 2 and priority of S 1 are the same and that is equal to the priority of T 2. I am sorry again priority ceiling of S 2 and priority ceiling of S 1. So, let me just cut it out a little bit, let me just say that ceiling of S 2 is the ceiling equal to the ceiling of S 1 and that both of them are the same as the priority of T 2.

Now, once again I bring into the picture the earlier example. So, first T 1 is started, and T 1 is starting from time t zero to T 1. So, here also we will do the same thing; T 1 is staring let us mark the time line first. So, this is t 1, this is t 2, this is t 3, this is t 4. So, initially T 1 starts and at T 1 time t 1 it asks for S 2 it asks for resources, so which is guarded by semaphore S 2. So, now, it is within the critical section, it is within the critical section now. So, this is within S 2. At this point T 2 time t 2, T 1 arrives, so T 1 arrives here. And T 1 having the higher priority will carry on up to t 3, I am sorry this is T 2, it will carry on up to time t 3. At this point, it has in the earlier case it is asking for resource S 1. So, at this point, it is for trying to perform P S 1.

Now, under normal circumstances as was in this case that is in the case of priority inheritance P S 1, it could continue right, it could continue and here those are change the priority inversion to place here, but now according to priority ceiling protocol before I allocate the resource S, I mean corresponding to S 1 to T 2 to this task T 2 I will have to check. What do I check? I will check the ceiling of the priority ceiling. So, I find that it can get the resource our rule is that it can get the resource provided there it sees priority is higher than the ceiling of all other tasks, but that is not the case. Here S 2 has being already entered and I means it is being used and the ceiling of S 2 and the ceiling of S 1 are the same is not greater. This one is not greater. Therefore, it cannot actually perform this P S 1, this one is stuck, this cannot be performed.

Instead, I will allow this one to go one; and may be some had down the line it asks for P S 1, it will get P S 1 here right because that one is waiting it will complete P S 1 here and then perform V S 1 here. And as it does V S 1 here then T 2 can again start executing this S 2 right, thereby we are avoiding the dead locks scenario.

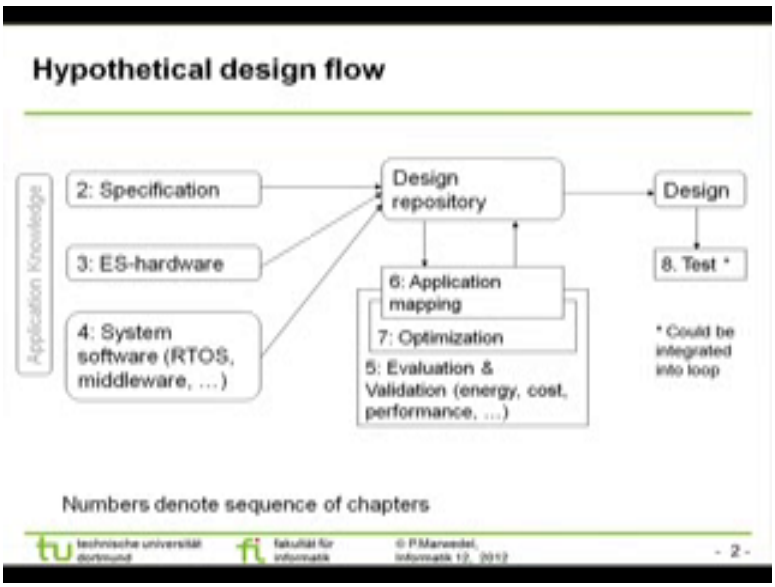
Student: S 2.

Yeah, it will this one will also do V S 2, this one will do V S 2, because it has completed both. So, the problem that was occurring earlier is that because of this reason it was going in right. And at this point, those are priority inversion occurring and it was rather priority inheritance of occurring, it was inheriting the priority it was inheriting the priority and was continuing, and here it was asking for the resource which is already captured, hence deadlock was occurred, you can see the circular waiting here. But in this case, what we are doing since we are putting in some priorities with the resources themselves. Now, this process T 2 is not being able to capture I mean here just because it has got higher priority, it cannot really continue because of the resource conflict. And the resource that is already some resource that is captured the ceiling of that is not lower than you. So, we will continue all right.

Student: Sir, last one the task S 1.

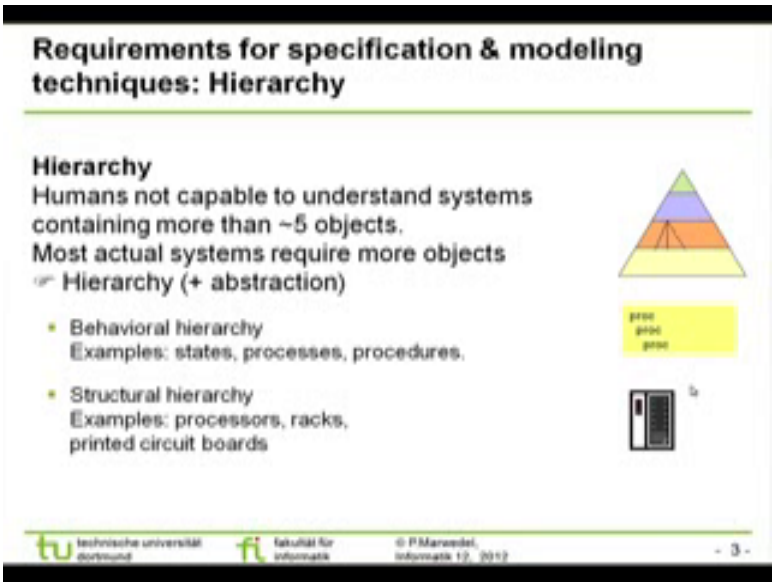
This will be S 1, this is S 1, so working on S 1 right, so that clarifies how the priority ceiling algorithm avoid dead lock as well as deals with priority inversion. Now, with that we conclude our discussion on real time scheduling, there are many more aspects and many more issue which we can take up later or you can study by yourself that will not form a part of this course any further. So, next we move to another topic that is very important that is modeling and specification of embedded systems.

(Refer Slide Time: 18:25)



First will be discussing about models, now, this was the original diagram that we had taken from Prof. Peter Marvedal and different tasks were being shown. We have just lets recapitulate we have discussed in detail along with quite a few tutorials the embedded system hardware aspect. Now, we have talked about the system software, real time operating systems we have talked about. Now, we are coming to specification we have coming to the specification little later, but that really does not matter.

(Refer Slide Time: 19:05)

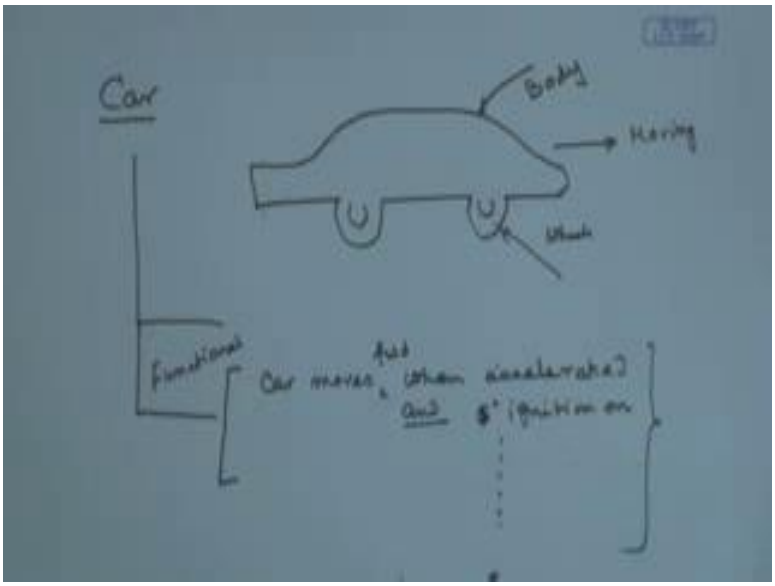


So, now when we talk about specifying an embedded system. One simple way is natural language. A normal English, we specify what we need, but what is the problem with that the problem is that often our conversational natural sentences are full of ambiguities. And because of those ambiguities if there been automated design environment, it will not be able to understand the correct semantics of the correct design intention. Therefore, we need some formal approaches. Now, the formal models, several models have been created for embedded system design, some of the properties which are desirable for this modeling, first is the hierarchy. It is said that also from the cognition point of view, if I go on telling you 5 more than 5, 5 digit numbers or plus minus 2, so 5 to 7 numbers each of 5 digits may be, after while after 5 such numbers we will not be able to recall them.

So, it is said that the systems becomes if we have more than 5 objects say in one particular model then the model becomes really incomprehensible. And actually most actual systems will require many more objects. If I have a complex circuit then that will

have number of transistors, number of gates registers are everything and so it is very difficult to understand, but instead if we can take them module wise that is why the moderate is there, but there is one thing in hierarchy besides modular. That means, we can present the system at different levels of abstract.

(Refer Slide Time: 21:44)



To start with let me before we talk about this let me tell you, what is a model. A model is nothing but representation of a miniaturized representation of the actual thing through some other language. Say, for some example you want to describe a car all right. Now, you can describe a car with something like this that is a car which has got all right this is a car looks like a car now like a, beautiful. Now what does it say and I can say that these are wheels, this is the body. So, this is at one level, how a car looks all right. I can also tell a car to be here I can also say moving like this. So, the car moves and car has wheels. I can further break down this structure and can show inside the of the car.

So, inside of the car I cannot draw, but I can further decompose this as I insight of the car and show the seats steering and all those things. So, these are the top level then I can go on decomposing this. I can also say, but whatever my model is it I can say that the ultimately the model must describe. So, this was the structural loop wise. Here I can also try to give a functional model. A functional model say that a car, number of things, car moves when accelerated right, is that ok? Natural language which I have done now moves, but provided the start is on etcetera and ignition on etcetera, etcetera, etcetera.

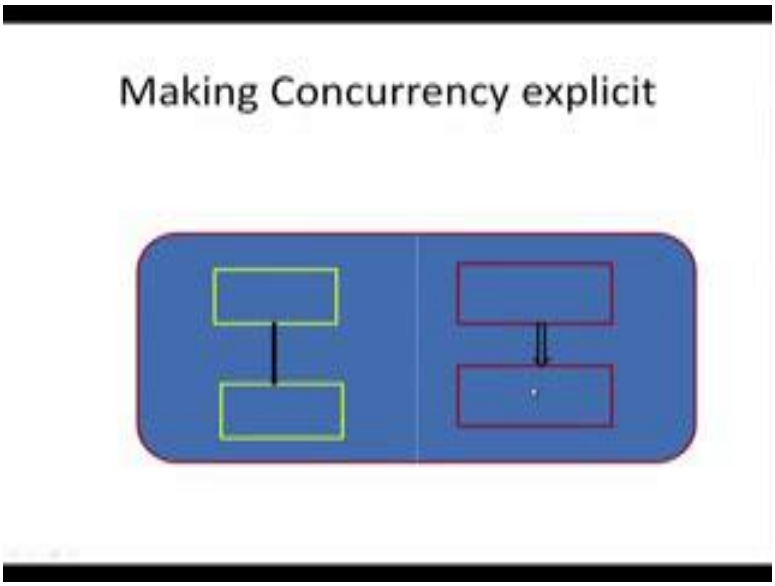
Now, while I go one describing this I may miss out on some points. If I had written here the car moves forward with this that would be wrong, because the gear thing also comes into picture whether it is rear or something. So, there number of issues that we will have to say.

Ultimately say for example, say now let us look at another thing see a model of a car versus model of a truck, how will that differ, it will be by the size, by the capacity, now there are many issues. So, ultimately a model will lead to a set of rules which describe the behavior I am talking about behavioral model, which will described the behavior of the system. And the system will follows such rules. When accelerated the car moves is a rule that holds for all cars, but that may not hold for may be some other object. So, in that way, a model should be comprehensive, a model should also show the functionalities very clearly in an unambiguous manner.

What we try is we have got some model of this system, but for our purpose, we will have to translates the model in to some executable form. Why, do we need such executable form? Because say here I have written certain thinks about this car. Is it complete, does it correctly describe the thing? If it could have translated that in the forms from program which could be executed, and I could have seen the result of that and see if my description is correct and complete. Therefore, often we desire the model that we choose for describing any embedded system to be executable that means, you should be able to execute it and see its performance.

With that let us come back to this. So, what I say it is just as I have said that the car could be seen at the top, at the outside or I can see the inside of the car, and then I can further detail I can go to the color of the seat covers and all those things. So, those are even lower granularity of details all right, and more and more detail which can come at different levels of abstract and that becomes easier for me to describe. There can be behavioral hierarchy now coming from the car to our embedded system scenario, the behavioral hierarchy can be states, I can say that their number of states or systems can be something like this.

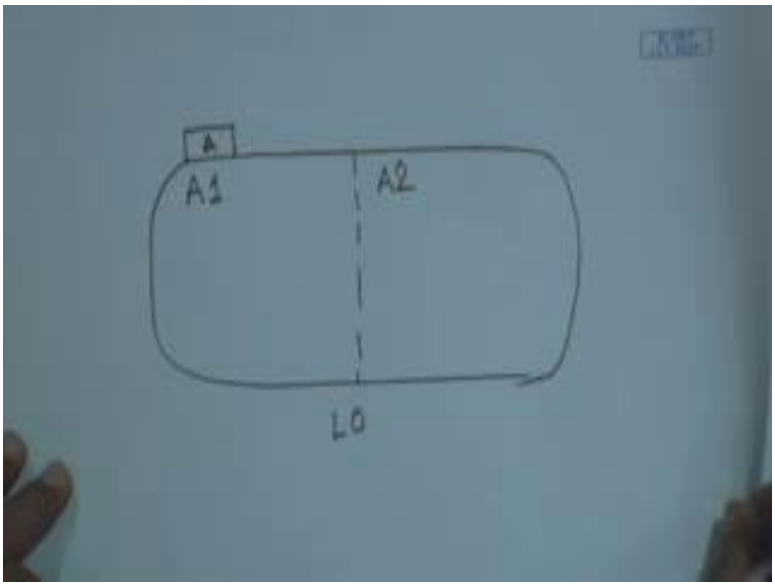
(Refer Slide Time: 27:08)



There are two states from this state it goes to this state; and from this state, it comes back to this state that is that my description. At another level, I can go to the processes and procedures that are there actually running when the system is in this state, what are the things that are happening so that is the level of abstraction and I want decomposing them at different levels of hierarchy. There can be similarly the structural hierarchy where I can have processes, racks, printed circuit board.

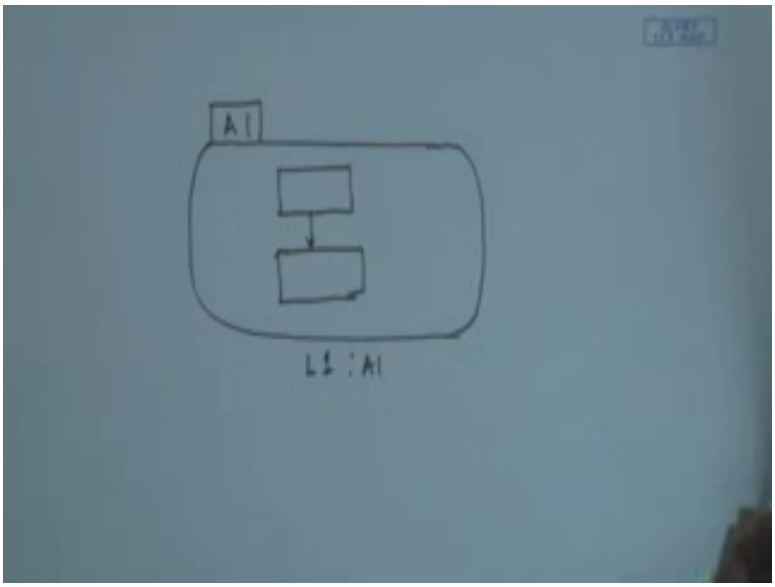
So, here for example, I can say this is a structural hierarchy at the top. Now, inside I can go in this switch box then there are racks and all those things I can describe. Another desirable, so hierarchy is one of the desirable properties. Another desirable property for any embedded system specification model is to make the concurrency explicit. So, here you see this white line in between shows the concurrency that this part of the behavior can run concurrently with this right side of the behavior; both this behaviors can run concurrently. Now, again you can see the hierarchy. At the top level, I say that this one if I had name this A 1 and this to be A 2, then A 1 runs concurrently with A 2.

(Refer Slide Time: 28:54)



So, in a hierarchical model, I should have first drawn it like this that there are some behavior which I am trying describe at it consist of two distinct behaviors A 1 and A 2 it just says that A 1 and A 2 are concurrent that is the level 0 of my description.

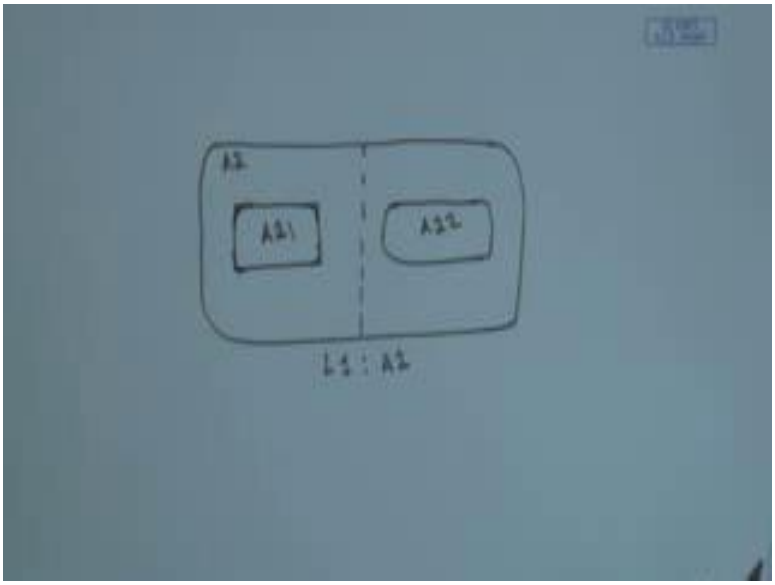
(Refer Slide Time: 29:32)



Next, I can come to A 1 I can say that this is A 1, now I could have said that this one is actually behavior A which consist of two concurrent behaviors A 1 and A 2. And here I can say that the behavior A 1 is actually as I have showing in the diagram there is some process followed by another process which just sequential may be right. So, you see this

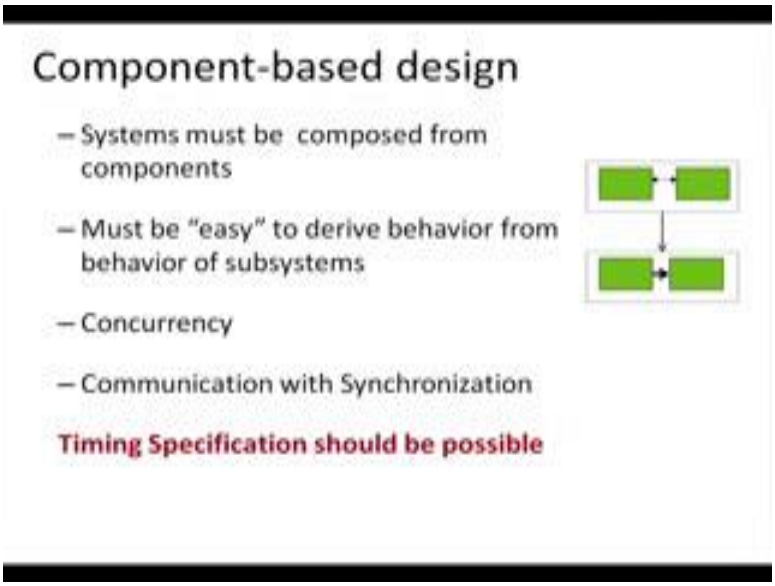
is the top level of hierarchy L 0 describing a and some concurrency is being shown here it is L 1 showing A 1, I can similarly show A 2 to be something like this.

(Refer Slide Time: 30:10)



Where further I can say A 2 is again concurrent behavior of may be A 2 1 and A 2. Tthis is level 1 of A 2 this is a little different from what I was showing in the power point. So, why I brought this n is that I can further show concurrency is down the line. So, coming back to the power point here that I was showing here is that therefore, here no concurrency is being shown A 1 and A 2. And at the next level, I am showing that A 1 is actually a sequence of these two yellow blocks, and this one is the sequence of this two red blocks.

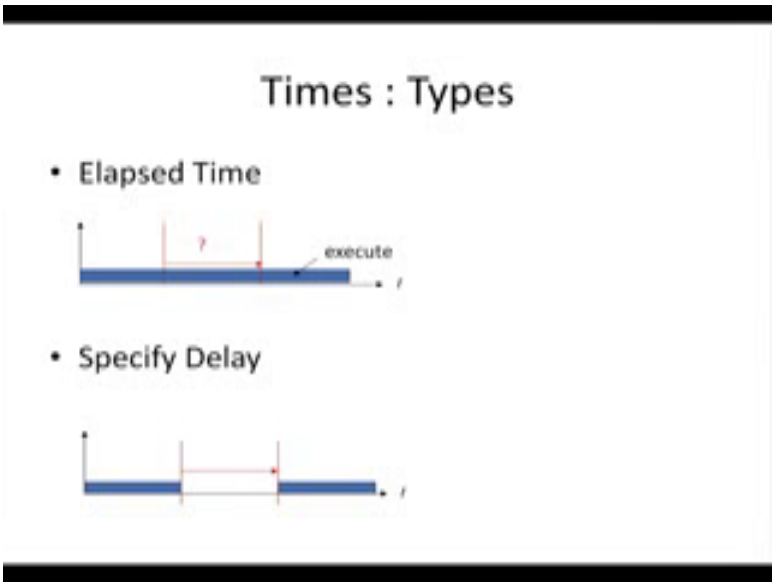
(Refer Slide Time: 31:17)



Now, when we try to do a model based design actually we are talking of some component based design. We are as I said that we break it down into lower levels of granularity. And the system is actually made of different components I can say at the top that this is the first dotted line is one component, and this dotted line is another component and these two components communicating among each other. So, the other important point is the communication between the component. Now, these components as you can see is hierarchical description, because this component can be further broken down into these two green components which have get their communication among themselves. And this component again is can be broken down these two green components, where there is a sequential communication between them.

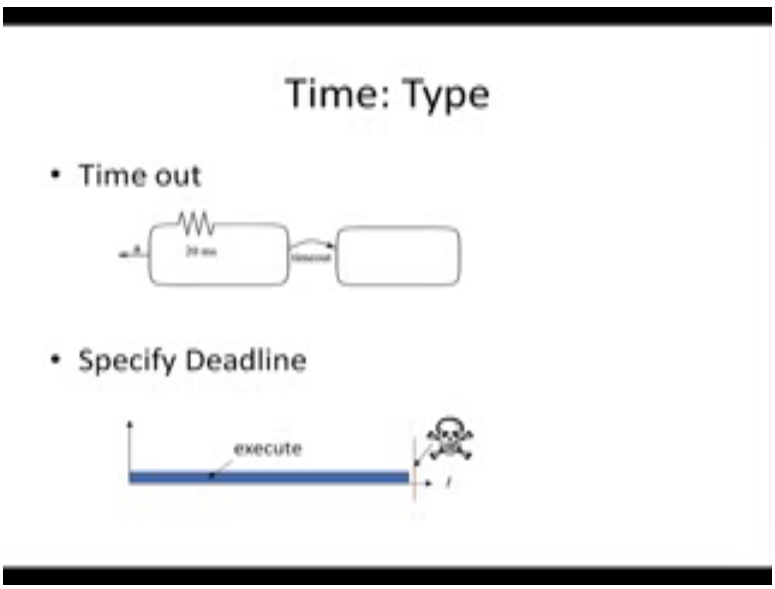
So, the entire behavior is consisting of components, so that is one important thing we should be able to show the concurrency that this one and this one. When this one is active this one might be active. Concurrency, we have to show and communication is another important point that we have to capture in any model. So, in the model, we have to capture first of all the components, their behaviors, and how they communicate. And wherever possible it is very much desirable although all modeling languages do not provide us that timing specification should be possible to be expressed.

(Refer Slide Time: 33:08)



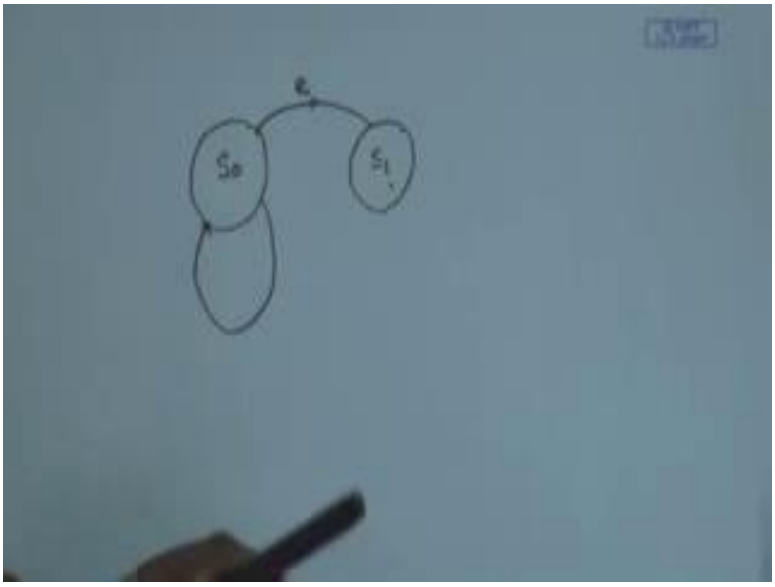
What are the types of times that we are interested in. We have already encountered when we discussed about real time systems, different forms of time that we are discussing. One is the elapsed time we start from any particular point, and see that from this time, this point onwards how much time has elapsed while a program is executed. We should be able to specify that we should be able to know that, we should be also be able to specify delays that the first run of this program, and the second run of this program should be delayed by a certain amount. After this completes, the first run, the second run should be delayed by this extent. So, delay is another type of time that we often will require.

(Refer Slide Time: 34:08)



Here is another one that is very important I was just now when I was drawing such diagrams like this, I did not mention, but these means as we will see later that this can be different tasks and these are also different processes. If I consider this to be processes, then suppose a task is here in this stage, and this state as I said will consist of number of processes and it can be executing within this process itself traversing different substrate of this. Here I specify a time out that time out here is shown to be 20 millisecond. Now all of you are aware of process finite state machine and all those.

(Refer Slide Time: 35:09)

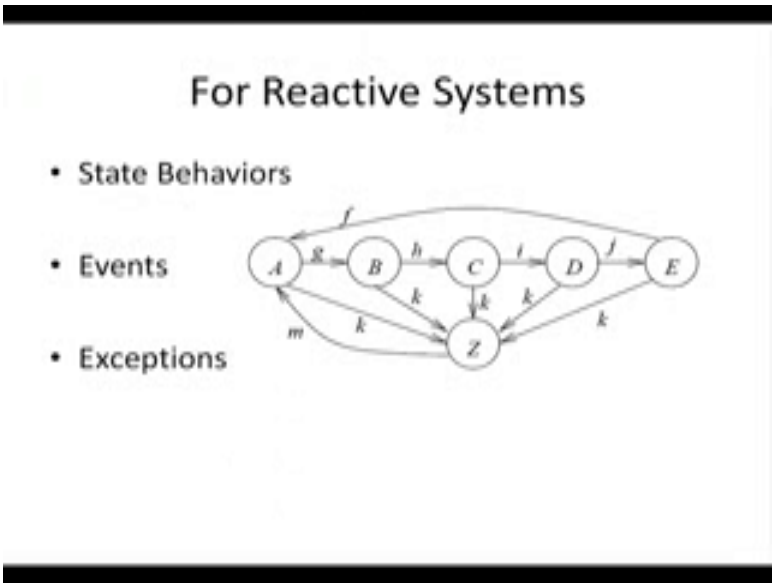


So, if I have a state S 0, from there I can exit and go to another state S 1 based on some event when some event takes place I go out from one state and enter another state or maybe some times I can remain in this state also that the different thing. So, here what is happening here is on an event a, it can go out, but if the event a does not take place within 20 millisecond then there will be a time out generated until go to some other state. So, timeout is another type of time specification not too many modeling languages provide that, but will see later that state chart allows such specification.

And this one is very common to us specifying the dead line with respect to the tasks, because we have seen that several times in our real time systems scenario that it must complete its execution within this time. So, when I was mentioning the desired it out what are desirable for a modeling language or modeling system or a embedded system, I must be able to break them up in different component in hierarchical manner. I must be

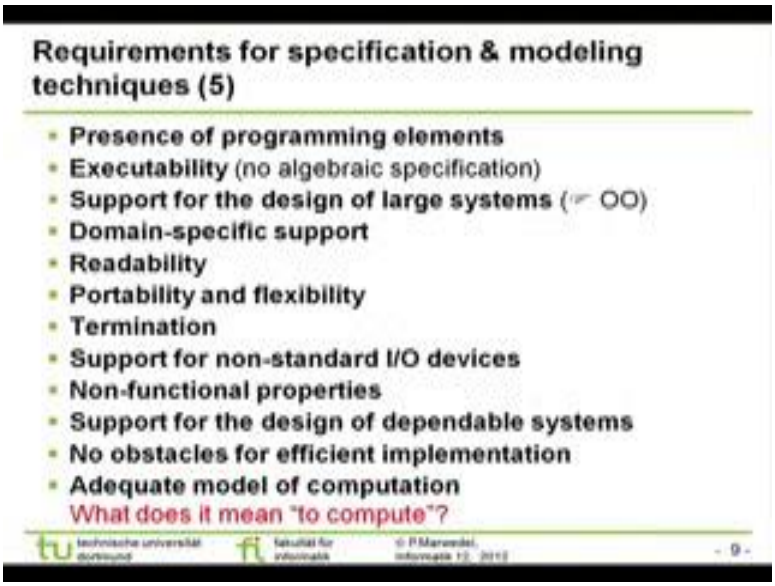
able to specify the concurrence, I must be able to specify the computation within the components will come to that we must be able to specify the communication, and it will be a nice if I can specify the timer.

(Refer Slide Time: 36:57)



So, here for reactive system, you all remember what a reactive system is a reactive system reactive certain events. So, I can describe this diagram will come time and again there number of states and based on different events g, h, i, j on these events is state transition taking place; on f it is coming back to this state is just an example. And so there can be state behavior each of this states can have its own behavior all right is behaving in a particular way doing in a type of computation. And when the state transition takes place on an event, I may change the behavior, I may try to start doing some other tasks. And also there can be exceptions, say for example, here it is shown that for all these cases on the exception k, we are coming to another state Z. So, exceptions some specific error condition or special conditions should also be we should be able to specify that.

(Refer Slide Time: 38:12)



So, I will conclude with this the requirement for specification in modeling technique are there should be now this one I have not discussed in detail presence of programming elements by that I should be able to express the behavior. I have said that there should be some state and a state has a behavior, but I must able to specify what is done there; although we are not restricting ourselves to the typical imperative programming like C, C plus plus etcetera, because it has got several problems if I solely depend on that.

If I only depend on such procedural languages, then there will be a number of each of these tasks will be different threads, and all the associated problem of synchronization, deadlock all those things, race condition will come in and we need support for that. So, whether we are facing such problem or not is easier to be visualized if I go for some other modeling. And when I come to a real lower granularity of behavior or task description what is to be done there I can take help of programming language will see one such method later.

Now, I have already said executability that I have done some specification is it ok? How do I know I should be able to execute it? Support for the design of large systems like object orientation or different levels abstraction object orientation also does provides you the abstraction. Specific domain support, the specification must be readable goes without saying, portability, termination must be specified. Now, it should be easily

implementable and there should be adequate model of computation, adequate model of computation.

I stop here today with this question, what it means to compute in this environment in this context, when we want to have an adequate model of computation, what does it mean? Because that must be captured in my specification language, so we will take it up in the next lecture.