Embedded Systems - 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 – 03
Supervisory Control Theory of Discrete Event Systems

Welcome. In this lecture, we continue our discussion on the Modeling of Embedded Systems. In the last lecture, we discussed two principle strategies for modeling embedded systems; namely the sequential program model and the finite state machine based model. We also discussed three extensions to the finite state machine model; namely the extended finite state machine or the finite state machine with data path and then we discussed hierarchical or concurrent finite state machines HCFSM, in which a state of the FSM could also incorporate another finite state machine. And, we discussed mechanisms for expressing concurrent processes and also hierarchical structure in an embedded systems; how to model such concurrent processes and hierarchical processes we understood and saw in the last lecture.

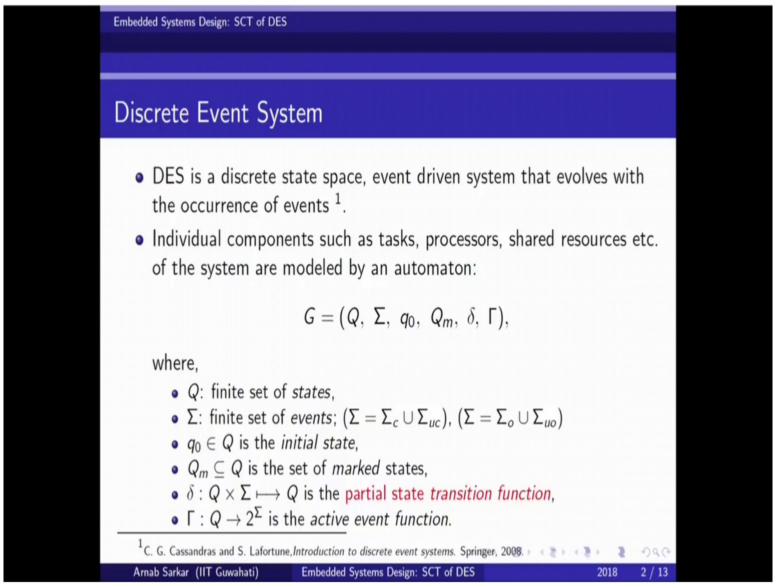
We saw two types of decomposition the or decomposition for hierarchical for hierarchical or type decomposition and the and decomposition where we could describe or specify concurrent processes. Then we also discussed the program state machine in which each state of a hierarchical it was an extension of the hierarchical and concurrent finite state machine. And this allowed each state of the hierarchical and concurrent finite state machine to possibly include sequential programs. So, the description of a state could now include sequential programs.

In today's lecture, we discuss a slight deviation of another slight deviation of the finite state machine modeling strategy called discrete event systems. In fact, discrete event systems is not only a in strategy, but also allows automatic and correct by construction synthesis of the controller of the embedded control task or the embedded controller that we want to design. So, this is a strategy which not only allows modeling, but also automatic synthesis of the embedded controller that we intend to design.

However, in order to do such synthesis we need to be able to accurately and formally model each component on the embedded system using the discrete event systems or

DES. And if we can model the in all components of the system the different processes or functionality is the different resources; all these the behaviour of all these components if they can be accurately and completely and formally modelled then we have a methodology for automatic and correct by construction synthesis of the embedded controller. So, let us discuss this extension of finite state machine called discrete event systems.

(Refer Slide Time: 03:52)



Embedded Systems Design: SCT of DES

Discrete Event System

- DES is a discrete state space, event driven system that evolves with the occurrence of events [1].
- Individual components such as tasks, processors, shared resources etc. of the system are modeled by an automaton:

$$G = (Q,\ \Sigma,\ q_0,\ Q_m,\ \delta,\ \Gamma),$$

where,

- Q: finite set of *states*,
- Σ: finite set of *events*; $(\Sigma = \Sigma_c \cup \Sigma_{uc})$, $(\Sigma = \Sigma_o \cup \Sigma_{uo})$
- $q_0 \in Q$ is the *initial state*,
- $Q_m \subseteq Q$ is the set of *marked* states,
- $\delta : Q \times \Sigma \longmapsto Q$ is the partial state *transition function*,
- $\Gamma : Q \rightarrow 2^{\Sigma}$ is the *active event function*.

[1] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 2 / 13

A discrete event system DES is a discrete state space, similar to FSM because it has a finite set of states, is a discrete state space even driven system that evolves with the occurrence of events. So, there is a change in state when events occur individual and in this system we the supervisory control. So, a discrete event system is a discrete state space similar to FSMs, as FSMs have finite states this also has a finite set of state. So, it is a great state space even driven system that evolves with the occurrence of events.

So, there is a change in state when events occur. All components of the system has to be first model using such a discrete event system which changes states with the occurrence of events. So, individual components such as functionalities or tasks, resources or processors other shared resources etcetera of the system are modeled by an automata. Now, before going into these automata so, we first discussed the title of this lecture was supervisory control of discrete event systems. So, what is supervisory control of discrete event systems? The intention of these processes of this modeling methodology modeling in this way as I told you is ultimate synthesis of the controller.

This controller is called the supervisor because it controls how it controls the behaviour of the embedded system in a way we want it to behave. So, that it does not behave in a way that we do not want. So, the behaviour is restricted, the unrestricted behaviour is controlled by the supervisor and how does it do? The supervisor enables or disables certain events at each state. So, it will disable or enforce certain events or it will or it will not allow certain events and by that way it will control what the embedded system can do and what it cannot do, ok.

With this introduction we go into the formal definition of a discrete event system. So, discrete event system is a six tuple has six tuples Q, sigma, q 0, Q m, del and sigma; where Q is a finite set of states like in an FSM, sigma is a finite state set of events: this finite set of events includes both controllable and uncontrollable events. So, you if you see this sigma consists of c sigma c, union sigma uc. Here sigma c denotes the set of controllable events; that means, the supervisor can control or prevent the occurrence of these events. These are controllable by the supervisor. So, the occurrence of these events can be prevented by the supervisor if it so wants and it also includes union sigma uc, the system will also include uncontrollable events. And these uncontrollable events cannot be controlled by the supervisor the occurrence of these cannot be controlled by the supervisor, ok.

There are certain events so, sigma another class. So, we can classify the events in terms of sigma c, union sigma uc. Similarly, we can also classify the events as sigma o sigma union sigma uo. So, here sigma o is the set of observable events those events that can be observed externally the occurrence of these events can be observed externally and there can be certain events that cannot be observed from the outside world. So, the set of events will include sigma o, union sigma uo, q 0 small q 0 belongs to capital Q is a special event called the initial state. The entire embedded system this in whatever we are modeling a particular component if it is the component will start from this initial state. For example, in our earlier lift controller the idle state was the initial state of our system at for the elevator.

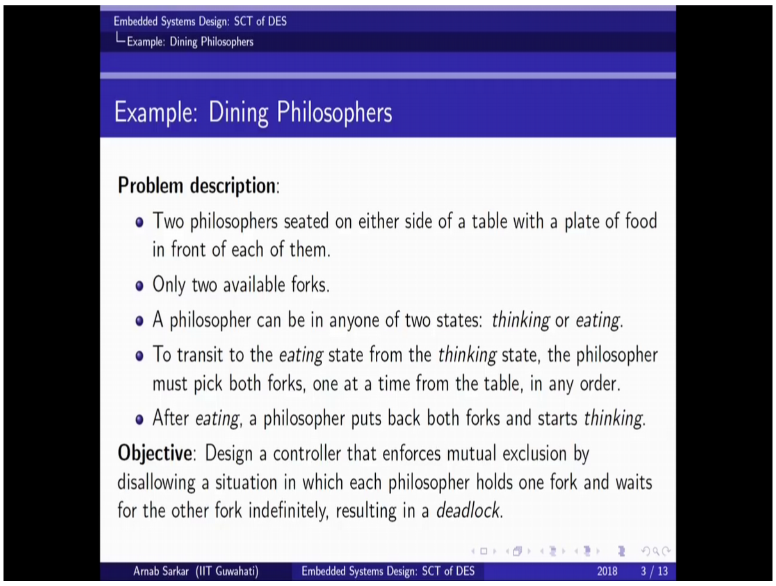
Capital Q m belongs to Q it is a subset of Q sorry capital Q m is a subset of Q is the set of marked states and it is a subset of states within q what do these states specify? These are the states at which the component the behaviour of the component can legally complete. So, these are the completion states for the legal completion states for the embedded control for the system that we are modeling. So, Q m denotes the set of states in which the component can complete its execution, ok.

Del is the transition function. So, del is the transition from a function from a state within Q capital Q and on a particular event from sigma. So, at a given state for a given input for a for a given input event, so, I have an event that has occurred at a given state and on that event I go to the next state which also is a member of capital Q. So, capital Q cross capital sigma goes to Q is the partial state transition function. So, here is the difference from finite state machine, it is not a total transition function; that means, that we do not need to have definition for the transition on all events in sigma.

So, on which events in sigma do we need to have this definition of the transition at a given state? It is all those it is for all those events that are defined in gamma which is next. So, gamma is gamma is a part of Q and it is called the active event function. So, there are a set of states called the active event set. Meaning that within any given particular state of capital Q within capital Q I take a state and there I will have an active event set; that means, those events from capital sigma which are legally possible at that particular state. Not all events in sigma will be legally possible at that particular state within the system that we are describing.

We will take an example, but this is what it is. So, therefore, del will be defined for only the active event set at a given state. So, a given state has an active event set which is a subset of the sorry it has an active event set which is a subset of the events or in sigma and on those events the del function or the transition function will be defined.

(Refer Slide Time: 11:37)


Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

Example: Dining Philosophers

Problem description:

- Two philosophers seated on either side of a table with a plate of food in front of each of them.
- Only two available forks.
- A philosopher can be in anyone of two states: *thinking* or *eating*.
- To transit to the *eating* state from the *thinking* state, the philosopher must pick both forks, one at a time from the table, in any order.
- After *eating*, a philosopher puts back both forks and starts *thinking*.

Objective: Design a controller that enforces mutual exclusion by disallowing a situation in which each philosopher holds one fork and waits for the other fork indefinitely, resulting in a *deadlock*.

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 3 / 13

Now, we will take first an example of an embedded system and we will also understand what we want to design, ok. I will take the example of the very popular dining philosophers problem. And then, we will show how automated synthesis from that how individual models can be developed for this embedded system and then how automated sis synthesis of the embedded controller can be designed for it.

So, now let us first describe the system the embedded system that we have DES that we intend has two processes and these two processors these two processes are functions we name philosopher 1 and philosopher 2 and there are therefore, these two philosophers are two processes in the embedded system that we two processes are functionalities in the embedded system that we want to design two philosophers are seated on either side of a table. So, this is how it is defined, two philosophers are seated on either side of a table with a plate of food in front of them. So, these two philosophers are two processes and the two plates of food are the input data that these processes will consume.

So, two philosophers are seated on either side of a table, these are two programs or processes and there is a plate of food that this program will execute with, on in the path in the front of each of them, these are the data the plates of food are data. Only two available forks; so, these forks are resources, but we will talk about the resources a bit later. So, what are these forks we can enforce, but we can understand what these forks are. These forks are the processing resources based on which the philosopher will execute it is function.

So, the process philosopher will execute using the forks. So, it will execute using the processing resources and what will it do the process is of eating. So, the processing the that the processing that you will it will do using the resource fork is eating of what the data which is the food. So, we have only two available forks, a philosopher can only be in either two states. It is the philosopher is either thinking or it is eating, and to transit to the eating state from the thinking state the philosopher must pick both forks one at a time from the table in any order.

Now, how can the philosopher transit from thinking to eating, what will be the events? So, initially it will be in the thinking state and then it will pick one fork. So, the event will be the picking of the fork so, taking a resource and it will have to take both resources both the forks in order to be able to eat and this consumption of the resource this taking of the resource that will happen one after another not together. So, to transit to the eating state from the thinking state the philosopher must pick both forks.

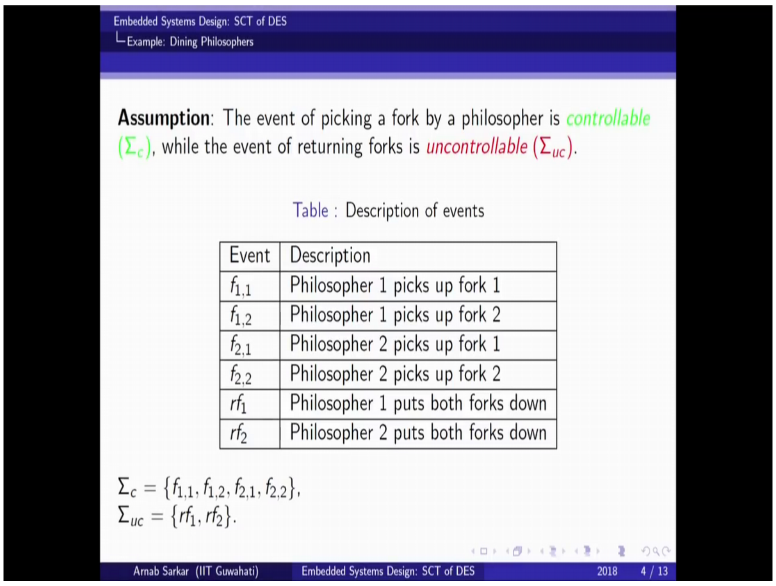
It requires both the resources one at a time from the table in any order. After eating a philosopher puts back both forks and starts thinking. So, after eating is complete after the processing is done it will relinquish the resources. Firstly, it captures the resources and eats and then after eating is done it relinquished the computing resources together, ok, both forks and then it starts thinking and goes back to it is initial thinking state.

Now, what given this so, I in this embedded system I have two concurrent processes two processes that that will run and these are two philosophers which wants to eat food placed in terms or placed in front of them each of these philosophers are can be in two states either thinking or eating in order to eat they will have to pick both forks eat. So, to philosophers both the philosophers cannot eat simultaneously right. So, the objective of the embedded controller is exactly to enforce this.

Design a controller that enforces mutual exclusion by disallowing a situation in which each philosopher holds one fork and waits for the other fork indefinitely. Now, we said that philosopher will pick the forks one at a time. So, a situation can therefore, happen that let us say philosopher 1 has picked one fork for eating and at the same time philosopher 2 also wanted to eat and therefore, it picked the other fork and then what happens is that both the processors both the philosophers are waiting for the other fork and nobody can start eating because the other fork or the other resource is taken by the other philosopher.

So, the objective of the embedded controller is to enforce mutual exclusion by disallowing a situation in which each philosopher or each process holds one fork or takes one resource and waits for the other resource in indefinitely leading to a deadlock. With this definition let us go about the modeling.

(Refer Slide Time: 17:30)



Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

Assumption: The event of picking a fork by a philosopher is *controllable* (Σ_c), while the event of returning forks is *uncontrollable* (Σ_{uc}).

Table : Description of events

Event	Description
$f_{1,1}$	Philosopher 1 picks up fork 1
$f_{1,2}$	Philosopher 1 picks up fork 2
$f_{2,1}$	Philosopher 2 picks up fork 1
$f_{2,2}$	Philosopher 2 picks up fork 2
rf_1	Philosopher 1 puts both forks down
rf_2	Philosopher 2 puts both forks down

$\Sigma_c = \{f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}\}$,
$\Sigma_{uc} = \{rf_1, rf_2\}$.

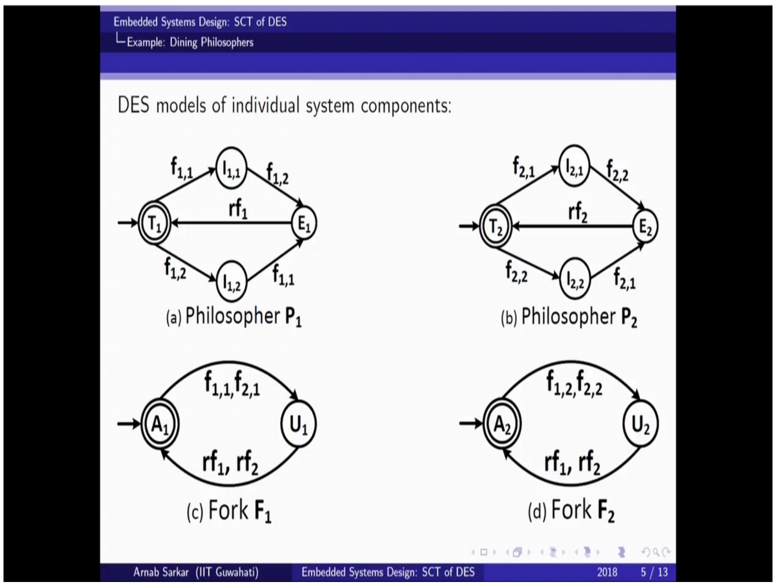
Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 4 / 13

Firstly we define the events the event of picking a fork by a philosopher is controllable, ok. So, the supervisor can control whether a philosopher can pick a fork, can take a resource at a given time or not. So, it will allow or disallow a philosopher or a process to be. So, that it will be able to pick a resource or not while the event of turning forks is uncontrollable. So, after eating it the supervisor cannot control it from relinquishing the resource, from leaving the resource right, ok.

So, what are the descriptions of events? So, we now symbolically describe the different events and name the different event symbolically. So, the first event is f 1, 1. So, f 1, 1 means philosopher one picks up fork 1; f 1, 2 means philosopher 1 picks up fork 2; f 2, 1 means philosopher 2 picks up fork 1; f 2, 2 means philosopher 2 picks up fork 2; rf 1 means philosopher 1 picks the puts both forks down rf 1 relinquishing fork 1. So, re sorry relinquishing forks by philosopher 1 so, rf 1 means philosopher 1 puts both forks down and rf 2 means philosopher 2 puts both forks down.

Now, whatever the set of controllable events in this sigma c sigma c are f 1, 1, f 1, 2, f 2,1, f 2, 2. So, the picking of the forks either by philosopher 1 or philosopher 2 is controllable by the controller controllable externally controllable by external entities which can control the embedded system and the occurrence while the occurrence of the relinquishing of the resources fork 1 fork 2 is uncontrollable. So, sigma uc is rf 1 and rf 2.

(Refer Slide Time: 19:39)


Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

DES models of individual system components:

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 5 / 13

Now, before the design begins we first model each of the components in the embedded system using the discrete event system model. So, first philosopher 1 so, what does it say. Firstly, the initial state is P 1 that is philosopher 1 is in the thinking state. When after some time let us say philosopher 1 wants to eat and therefore, let us say he picks fork 1. So, therefore, the event is f 1, 1 philosopher 1 picks fork one when he does this it goes to the intermediate state I 1, 1 in this state philosopher 1 has picked up fork 1 and then off after this intermediate state philosopher 1 will then pick the other form.

So, at I 1, 1 it he can pick fork 2, which is f 1, 2. So, philosopher 1 picks fork 2 and after this he has picked both forks and he goes to the eating state E 1. After eating he both forks using rf 1 and goes back to thinking state T 1. Similarly, if he if he picks fork 2, so, f 1, 2 so, at the thinking state he could have first picked fork 2. So, from T 1on event f 1 to it would go to state I 1, 2 which is another intermediate state in which. So, what is this intermediate state I 1, 2? Philosopher 1 has picked fork 2 and has not picked fork 1, that is I 1, 2. After he picks up fork 1 at I 1, 2 he goes again to the eating state E 1. So, philosopher 1 now again has both forks and he is eating after eating is complete, he relinquishes both the forks and goes back to the thinking state. This is the behaviour of philosopher 1.

The similar behaviour also goes for philosopher 2. So, either he is initially in the thinking state T 2 and then he can pick let us say fork 1 which is f 2, 1. So, philosopher 2 picks fork 1 f 2, 1 and goes to the intermediate state I 2, 1. So, this is the intermediate state in which philosopher 2 has picked up fork 1 and has not picked a fork 2 and then he picks up the other fork and goes to eating in a very similar as that for philosopher 1.

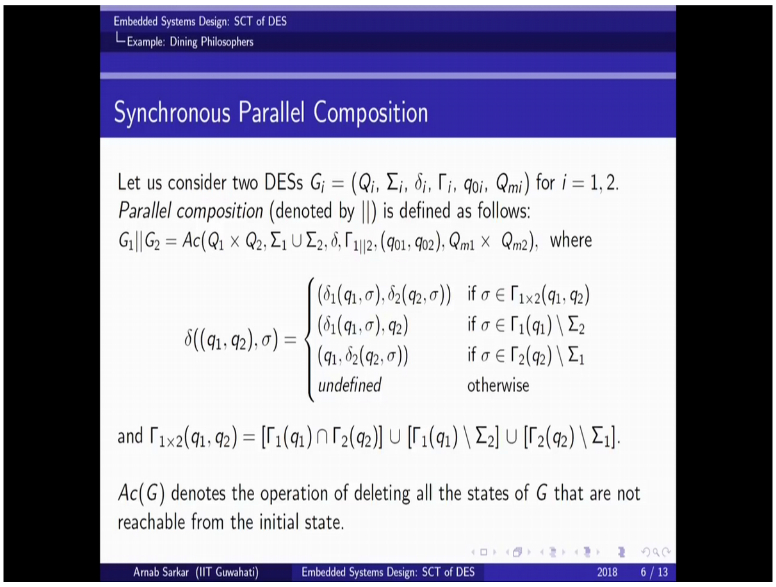
Now, how does the resources behave? So, this is how the two processes in our embedded system behaves. How does the resource what how do we model the behaviour of the resources? The resources can either been available state or in the used state. So, fork 1 that is the resource 1 is first in the initially in the available state A 1 and how can it go to the U state U 1 either through the event f 1, 1 meaning that philosopher 1 picks up fork 1 or through the event f 2, 1 in which philosopher 2 picks up fork 1. So, in either of these two ways it goes from the available state to the used state.

Now, after U 1 it goes back to the available state either on event r 1, rf 1 or rf 2. So, either philosopher 1 relinquishes the resource fork 1 or philosopher 2 relinquishes the resource fork 1, in either way it goes from the U state back to the available state. And, this is the same the same similar behaviour also holds for fork 2. So, it is in the initial state a 2. Now, what are the events that it can it that can take it to the U state? Either philosopher 1 or philosopher 2 picks up fork 2. So, f 1, 2 philosopher 1 picks up fork 2 or f 2, 2 philosopher 2 picks up fork 2. In either way it goes from available state to the used state. At the used state after eating rf it goes back to the available state through the events are f 1 or rf 2. So, fork 2 is relinquished by either philosopher 1 or philosopher 2 in either case it goes from U state back to available state.

Now, how do we model the concurrent behaviour of different components in the embedded system. So, before we go about the design of the controller we have to define the concurrent co execution behaviour of the different components of the embedded system in order to obtain the overall behaviour of the embedded system. First we are modeling. So, what are we modeling? The behaviour of the embedded system first we are going bottom up first we have modelled the individual components philosopher 1, philosopher 2, resource 1, resource 2, now slowly we have to obtain the concurrent co execution uncontrolled concurrent co execution behaviour of the entire embedded system on which control needs to be imposed.

So, how do we get the concurrent behaviour of such an embedded system modelled component component wise using DES to the composition of the DES through parallel composition. So, concurrent behaviour is modelled through the parallel composition of DES.

(Refer Slide Time: 25:43)



Embedded Systems Design: SCT of DES
Example: Dining Philosophers

Synchronous Parallel Composition

Let us consider two DESs $G_i = (Q_i, \Sigma_i, \delta_i, \Gamma_i, q_{0i}, Q_{mi})$ for $i = 1, 2$.
Parallel composition (denoted by $||$) is defined as follows:
$G_1||G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, \Gamma_{1||2}, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$, where

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Gamma_{1\times 2}(q_1, q_2) \\ (\delta_1(q_1, \sigma), q_2) & \text{if } \sigma \in \Gamma_1(q_1) \setminus \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Gamma_2(q_2) \setminus \Sigma_1 \\ \textit{undefined} & \text{otherwise} \end{cases}$$

and $\Gamma_{1\times 2}(q_1, q_2) = [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) \setminus \Sigma_2] \cup [\Gamma_2(q_2) \setminus \Sigma_1]$.

$Ac(G)$ denotes the operation of deleting all the states of G that are not reachable from the initial state.

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 6 / 13

So, let us consider two DESs G 1 that is G i. So, the first DES G 1 is Q 1, sigma 1, del 1, gamma 1, q 01 and Q m1 and similarly I have G 2 which is Q 2, gamma sigma 2, del del del 2, gamma 2 and then q 0 2 and Q m 2. The parallel composition of these 2 is defined as G 1 parallel G 2 equals to accessible part that A c, U c is the accessible part of Q 1 cross Q 2; that means, the comp the you have a composition of G 1 and G 2 what will be the number of possible number of states in the composed composite automata combining the behaviour of G 1 and G 2?

It will be the number of states in G 1 product the number of states in G 2. So, q 1 q 1, q 1 q 2, q 2 q 2, q 1 q 3. So, these will be the states in the. So, Q 1 of G 1 and let us say Q 1 of G 2 this can be one state of the composite embedded system. So, the product so, the number of states will be the product of the number of states in Q 1 with the number of states in. So, these are pairs of states of individual automata. So, this will be the number of states in the compost automata.

So, what will be the number of events sigma 1 union sigma 2? So, there could be some events that are not there in say G 2, those events will be there in the composed automata. The there can be some event which are there in G 2, but not which are dead in G 1, but not in G 2 that those will also be there in the composite automata and the common events of G 1 and G 2 will also be there in the compost automata. So, therefore, the events in the compost or composite automata are sigma 1 union sigma 2.

Then is the transition function which we describe next sigma 1 parallel to so, this is the active event set, we will we will just see that will also describe this next the initial states are the individual initial state. So, q 0 1 comma q 0 2 so, we say that the states are states of the composite automata are pairs of states of the individual automata right. So, what do we mean by this? What do we mean by the composite automata? We are saying that the composite automata, is composed of individual automatas running concurrently.

So, during the concurrent execution of the individual automata or processes at a at any given time let us say you have two automatas G 1 and G 2, G 1 will be at a certain state and G 2 will be in another certain state. So, if we consider that what will be the composite state it will be composed of that state where it is currently in G 1 and that state where it is currently in G 2. So, let us say it is in Q 1 of G 1 and Q 2 in G 2. So, q 1 q 2 q 1 comma q 2 is the composite state in which the overall embedded system is in at a given time at that given time. So, what is the initial state it is that pair state q 0 1, q 0 2 which consists of the initial state of both the automatas and the mark states again are the product of the mark states in both the automata Q m 1 cross Q m 2.

Now, we go into the transition function del mm the transition function del. Now, we will consider an example in which let us say I have a an even small sigma and the composite state in which currently of the composite automata is q 1 of G 1 and q 2 of G 2. So, the state of the composite automata is q 1 comma q 2 and we want to know what happens when this is this event small sigma occurs at the composite state q 1 comma q 2.

Now, there are three separate cases. That first the first case is that the first case is that this small sigma is in the active event set of active event set of both q 1 and G 1 and q 2 in G 2. So, if sigma is defined or sigma is part of the a given set of q 1 of G 1 and q 2 of G 2 then the first line of the on the RHS of this equation comes in. So, the next state is going to become del 1 q 1 sigma comma delta q 2 sigma. So, the composite state in which the automata will go from q 1, q 2 is that pair state which is defined by where q 1 would go on sigma and sorry, where G 1 would go on sing on this small sigma on from q 1 and where G 2 would go on small sigma from q 2 that pair state which is defined by what del 1, q 1 sigma comma del 2, q 2 sigma. So, it would go to that state. Why? Because, this small sigma is defined on both q 1 and q 2.

Now, let us say sigma is not defined in the entire behaviour of G 2. So, the second line comes the second line in the RHS says that when sigma belongs to. So, if you see in the RHS when this happens when if sigma small sigma belongs to gamma 1 q 1 that is it is in the active event set of G 1 at q 1, but it is not defined in G 2. So, it is not part of capital sigma 2, ok. So, if that happens then the next step; so, so if that happens. So, it is not defined in small in q 2 of G 2 because sigma small sigma is not defined in q 2 of G 2 it cannot go anywhere in from q 2 at G 2.

So, therefore, the next state will be del 1, q 1 sigma because it because small sigma is defined in q 1 of G one. So, it can move from q 1 to somewhere to the next to our next state from q 1 on sigma on small sigma. So, therefore, del 1, q 1 sigma is defined and it will remain in q 2 for G 2. So, the next state will become del 1, q 1 sigma comma q 2.

The third line the third line of the RHS happens for the opposite case. So, let us say small sigma is not defined as part of the active event set of q 1 in G 1, but small sigma is defined in the as part of the active event set of q 2 in G 2 therefore, in that case the next state would be q 2 comma del 2, q 2 comma sigma. So, it will remain in q 1 the automata the composite automata is going to remain in q 1 state of G 1 and it is going to move to the next state in G 2 from q 2 on the event small sigma. So, therefore, the next state becomes q 1 comma del 2, q 2 comma small sigma.

It is undefined otherwise. For example let us see what can be an undefined case. Let us say that small sigma is defined as part of let us say small sigma is defined as part of capital sigma 2; meaning that small sigma is a valid event in G 2. However, it is not defined at state q 2 of G 2 small sigma it is not defined at a state at the state q 2 of G 2; however, small sigma is a valid event in sigma 2. So, it is part of the active event set of some other state in G 2, it is not defined in q 2, but it is part of the active event set of some other state in G 2. It is not completely undefined. So, therefore, it is a part of capital sigma 2.

If that happens then the second line cannot take place the second line of the RHS the second line of the RHS in the equation cannot take place. So, del 1 q 1 comma sigma comma q 2 cannot happen if small sigma is part of capital sigma 2. Although it is not defined in q 2, ok. So, it is not defined in q 2, but it is part of the active event set of capital sigma 2, then the second line of the RHS cannot happen and it will become undefined, ok.

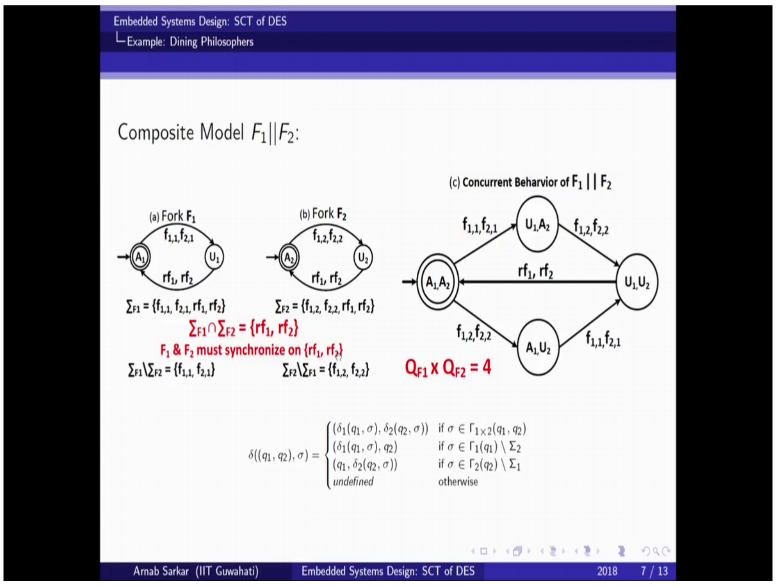
So, then comes what is capitals capital gamma 1. So, what do we mean by the active event set at q 1, q 2 of the composite automata. So, the active event set of q 1 of at the state q 1, q 2 of the composite automata is defined as follows. It is it is sigma 1, q 1 intersection sigma 2 q 2. So, whatever is the common active event at q 1 and q 2 those events are part of the active event set of q 1, q 2 first. So, it has it is composed of three components, right. So, union it has two union operators in operations in between. So, I am talking of the first part the first part sigma 1 q of the RHS the first part of the RHS what does it say sigma 1 q 1 intersection sigma 2 q 2, what does it mean? So, all the common active events at q 1 and q 2, so, those are part of the composite active event set of the composite state.

What else is part of the active event set of q 1 q 2? Those events that are defined as part of q 1 so, it is part of the active event set at state q 1 of G 1, but that those events are not defined in the entirety of G 2. Those events will also be included as part of the active event set of q 1, q 2 and then the opposite as well those events that are part of q 2 of G 2, but are not defined in the entirety of G 1. So, it is not part of capital sigma 1 those events will also be included in the common in the active event set of q 1, q 2.

Now, as I told accessible part what does this accessible part of, tell me; the accessible part of the composite automata. So, we say that G 1 parallel G 2 is the accessible part of the composition that we have obtained through this. So, what is the accessible part, denotes the operation of deleting all the states of G. So, after all this operation that we have got this composition operation we can get some states which are not reachable from the final from the initial state of the composite automata. So, those states which are not reachable from the initial state of the composite automata will be deleted. So, A c G denotes the operation of deleting all the states of G that are not reachable from the initial state of the composite automata.

So, this is the definition of synchronous parallel composition and with this definition we will see how we slowly build the behaviour the composite co executor behaviour of the embedded system that we are given; the embedded system consisting of two parallel processes p 1 and p 2 the two philosophers and the two resources f 1 and f 2.

(Refer Slide Time: 39:43)



Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

Composite Model $F_1||F_2$:

$\Sigma_{F1} = \{f_{1,1}, f_{2,1}, rf_1, rf_2\}$

$\Sigma_{F2} = \{f_{1,2}, f_{2,2}, rf_1, rf_2\}$

$\Sigma_{F1} \cap \Sigma_{F2} = \{rf_1, rf_2\}$

F_1 & F_2 must synchronize on $\{rf_1, rf_2\}$

$\Sigma_{F1} \setminus \Sigma_{F2} = \{f_{1,1}, f_{2,1}\}$

$\Sigma_{F2} \setminus \Sigma_{F1} = \{f_{1,2}, f_{2,2}\}$

$Q_{F1} \times Q_{F2} = 4$

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Gamma_{1\times 2}(q_1, q_2) \\ (\delta_1(q_1, \sigma), q_2) & \text{if } \sigma \in \Gamma_1(q_1) \setminus \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Gamma_2(q_2) \setminus \Sigma_1 \\ \textit{undefined} & \text{otherwise} \end{cases}$$

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 7 / 13

So, first we will see further resources because first we will try to compose two to the two resource models. So, we will try to find out what happens what is the possible concurrent behaviour and the states that are there that will be there in the concurrent behaviour of the two resources? What are the possible states that can happen when these two resources are being used concurrently? What are the possible states that can happen when these two resources are being used concurrently; this is what we will see.

So, the left one is fork 1 or resource 1 the right one is fork or resource 2 we have seen. So, what are the events what are the possible events for resource 1 or fork 1, f 1,1, f 2,1 rf 1 and rf 2; either philosopher 1 picks up fork 1 or philosopher 2 picks of fork 1 and then both I philosopher either philosopher 1 or philosopher 2 relinquishes fork 1.

Similarly, for fork 2 or resource 2 the events are f 1, 2, f 2,2, rf 1 and rf 2. Now, what are the common events what are the common events in f 1 of the two resources. So, sigma f 1 intersection sigma F 2 is rf 1 comma rf 2.

Hence in the during the transition if when defining the transition f 1 and f 2 must synchronize on rf 1 and rf 2 because rf 1 and rf 2 are defined as part of both automata, then they must synchronize. I cannot take the second line and third line of the RHS of the transition function defined in the earlier in in the in the earlier slide for the transition function only the top line will be applicable, the top line where I get where I go del 1 and del both are applied. So, if sigma belongs to gamma 1 cross 2 q 1 comma q 2 ok.

So, F 1 and F 2 must synchronize on rf 1 and rf 2 and what are the events that are defined in F 1, but not in F 2? f 1,1 and f 2,1 are defined in F 1 and not in F 2 and what are the events that are defined in f 2 and not in F 1? f 1, 2 and f 2, 2. So, the event of picking up the forks by picking up picking up fork 1 by philosopher 1 or philosopher 2 is defined in the resource model of a fork 1, but not there in fork 2. Similarly, the events of picking up fork 2 by philosopher 1 or philosopher 2 is defined in the resource model of f 2 and it is not part of f 1.

Now, the number of states in the composite model will be given by Q f 1 cross Q f 2 and is equal to 4 because fork 1 has two states and fork 2 has two states. So, the total number of states in the composite model will be Q F 1 cross Q F 2 equals to 4 and what are the states the initial state will be q 0 1 comma q 0 2 which is a 1 comma A 2, A 1 is the initial state in F 1 and A 2 is the initial state in f 2. So, the initially the composite automata is in state A 1, A 2 and because it is also the marked state it has two circles. So, mark states I forgot to tell previously marked states in the automata are represented by these double rounded circles, ok.

So, the automata finally, why are these double rounded circles there it says that it is initially in available state then it is in U state. And then finally, it will go back to the available state when process 1 will relinquish let us say any process will relinquish fork 1, in F 1 say in F 1 it goes from A 1 to U state. And then goes back after it is relinquished it goes back to A 1 and that is it is idle or initial state and also that it is final or marked state.

So, therefore, in the composite automata A 1, A 2 is the initial state and also the mark state. At A 1, A 2 either philosopher 1 or philosopher 2 will first pick up let us say fork 1. So, if it can take an event for the let us say it in the event that fork 1 is picked up the event fork 1 is picked up is represented by the arrow going from a 1, 1 to U 1 comma a 2, ok. So, when fork 1 is picked up either by fork either by philosopher 1 or philosopher 2 that can happen through the events either f 1,1 or f 2 1. If either of these two events occur fork 1 goes to the used state and fork 2 remains in available state. So, the state that it goes to is U 1 comma A 2.

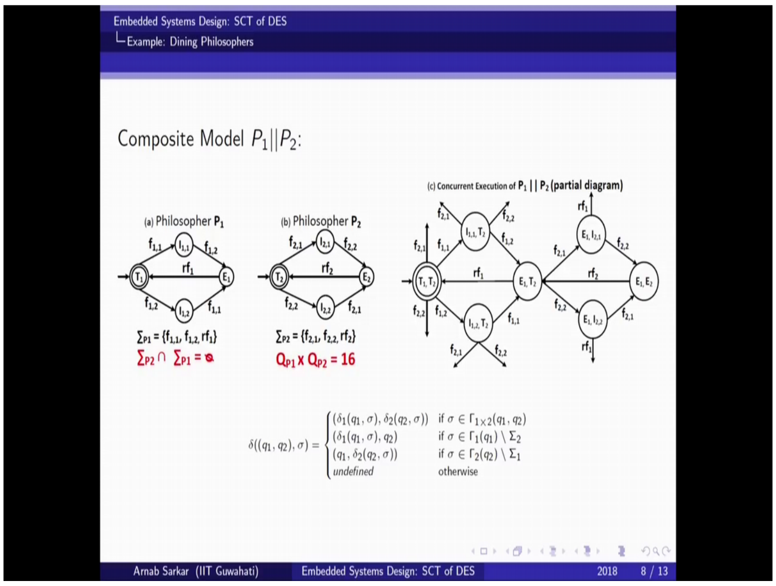
How does this happen because this let us say let us say f 1,1 has occurred at A 1, A 2. So, A 1, A 2 is the state and F 1 has occurred. So, del A 1, A 2 comma f 1,1 will be what will be del 1 A 1 comma f 1,1 comma A 2. Why? Because this event f 1,1 is defined as part of f o1ne, but it is not defined as part of F 2. So, capital sigma 1 contains f 1,1. So, capital sigma f 1 contains f 1, 1 but capital sigma F 2 does not contain f 1,1. So, line 2 of the transition function will be applicable.

So, therefore, A from A 1,1 in the composite automata from A 1, A 2 in the composite automata the next step that will be that that it the transition will happen on f 1,1 will be to U 1, A 2. Similarly, the transition can happen on f 2, 1. At U 1, A 2 what has happened fork 1 has been used, but fork 2 is available. Now, the next fork can be picked up the next fork will be picked up. When the next fork is picked up both the forks go to U state. So, at U 1, A 2 on event either f 1,2 or f 2,2 so, fork 2 is picked up either by philosopher 1 or philosopher 2 this happens and it goes to the composite automata goes to the state U 1 comma U 2.

Similarly, at A 1 comma A 2 I can reach the fork 2 can be picked first at A 1 comma A 2 fork 2 can be picked first by either philosopher 1 or philosopher 2. If this happens the lower edge will be taken from A 1 A 2 the lower edge will be taken and it the automata moves to A 1 comma U 2. So, from A 1 comma A 2 f 1 on the transition on the event f 1,2 or f 2,2 you move to the composite automata moves to A 1 comma U 2 in which fork 1 is in available state because it is not picked, but fork 2 is in used state and then from A 1 comma U 2 the other fork which is fork 1 will be picked either by f either by philosopher 1 or philosopher 2. So, from A 1 comma U 2 on the event f 1, 1 or f 2,1. I will again go to U 1 comma U 2.

Now, once I meant you I am I am in U 1 comma U 2 that is both my forks are in U state I will go back to A 1 comma A 2 either on rf 1 or rf 2. So, my composite automata will go back to the available state on the event rf 1 or rf 2. So, either philosopher 1 relinquishes fork 1 or philosopher 2 relinquishes fork sorry both the forks either philosopher fork 1 relinquishes both the forks or philosopher to re-link which is both the forks and then both the forks go from U state back to A 1, A 2 which is the initial and mark state. So, this represents the concurrent behaviour of the resources. Similarly, I can obtain the concurrent behaviour of the philosophers of the two concurrent philosopher's functions in my embedded system.

(Refer Slide Time: 49:28)



Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

Composite Model $P_1||P_2$:

$$\delta((q_1,q_2),\sigma) = \begin{cases} (\delta_1(q_1,\sigma), \delta_2(q_2,\sigma)) & \text{if } \sigma \in \Gamma_{1\times 2}(q_1,q_2) \\ (\delta_1(q_1,\sigma), q_2) & \text{if } \sigma \in \Gamma_1(q_1) \setminus \Sigma_2 \\ (q_1, \delta_2(q_2,\sigma)) & \text{if } \sigma \in \Gamma_2(q_2) \setminus \Sigma_1 \\ \textit{undefined} & \text{otherwise} \end{cases}$$

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 8 / 13

Now, what do I want to obtain the concurrent behaviour of the two processes of the two individual functions in the embedded system this is what I want to obtain. So, these two philosophers we have already modelled and seen how they are modelled and in this 1 the point to note is that sigma P 1. That means, the active the act the sorry the event set the total set of events that are possible in P 1 are f 1, 1 f 1,2 and rf 1. So, f 1,1 that is philosopher 1 picks up fork 1, f 1,2 that is philosopher 1 picks up fork 2 and rf 1 that is philosopher 1 relinquishes both the forks these events are defined in P 1.

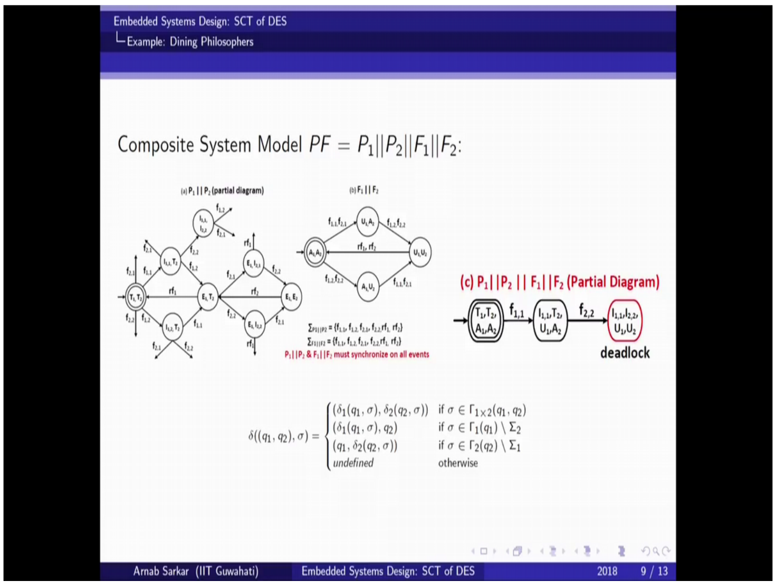
What are the events that are defined in P 2 what are the events f 2, 1, f 2, 2 and rf 2 that is philosopher 2 picks up fork 1 or philosopher 2 picks up fork 2 and philosopher 2 relinquishes both the forks. So, and there are no common events. So, they know they do not need to synchronize on any events. Why? Because sigma P 2 intersection sigma P 1 is 5 so, there are no common events as part of this one sigma P 1 intersection sigma P 2. What will be the number of states in the composite automata 16 because there are 4 states in P 1 and 4 states in P 2. So, 4 cross 4 is 16 and this is a partial con concurrent execution model for P 1 and P 2 together because the number of states are large 16. So, we have shown this contrary behaviour here.

So, let us take one or two states and how this happens here. So, initially the concurrent automata will be in T 1 comma T 2. So, both philosophers are in thinking state. Now, from this thinking state so, the program for T 1 let us say the program for philosopher 1 mm that they open that picks up or asked for the resource or takes up the resource for 1. So, philosopher 1 picks up fork 1, then what happens? When philosopher 1 picks up fork 1 we go to the state I 1, 1 comma T 2.

So, for the first automata philosopher 1 goes to the intermediate state at which it has picked up fork 1 and has not picked up fork 2 and philosopher 2 is still in the thinking state. Why this is feasible? Because f 1,1 is not defined as part of the behaviour of philosopher 2 because f 1,1 is not defined. So, line 2 of this definition of the transition function line 2 of the RHS of the definition of the transition function can be applied and by that we move from T 1, 2 T 1 comma T 2 to I, 1, 1 comma T 2.

And, then if philosopher 1 let us say happens to pick both the other fork as well that is on event f 1 comma 2 then I move from T 1 comma T 2 that is philosopher 1 moves to eating state and philosopher 2 remains in thinking state, ok. So, likewise we get the composite model for the philosophers for the concurrent behaviour concurrent co-execution behaviour of P 1 and P 2 by the composite automata P 1 parallel P 2 whose partial diagram is shown here.

(Refer Slide Time: 53:19)


Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

Composite System Model $PF = P_1||P_2||F_1||F_2$:

(a) $P_1 || P_2$ (partial diagram)
(b) $F_1 || F_2$
(c) $P_1||P_2 || F_1||F_2$ (Partial Diagram)
T_1,T_2,A_1,A_2
$f_{1,1}$
$I_{1,1},T_2,U_1,A_2$
$f_{2,2}$
$I_{1,1},I_{2,2},U_1,U_2$
deadlock

$\Sigma_{P1||P2} = \{f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, rf_1, rf_2\}$
$\Sigma_{F1||F2} = \{f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, rf_1, rf_2\}$
$P_1||P_2$ & $F_1||F_2$ must synchronize on all events

$$\delta((q_1,q_2),\sigma) = \begin{cases} (\delta_1(q_1,\sigma), \delta_2(q_2,\sigma)) & \text{if } \sigma \in \Gamma_{1\times 2}(q_1,q_2) \\ (\delta_1(q_1,\sigma), q_2) & \text{if } \sigma \in \Gamma_1(q_1) \setminus \Sigma_2 \\ (q_1, \delta_2(q_2,\sigma)) & \text{if } \sigma \in \Gamma_2(q_2) \setminus \Sigma_1 \\ \textit{undefined} & \text{otherwise} \end{cases}$$

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 9 / 13

Now, given the composite model for both the processes and both the resources so, I have P 1 comma P 2 partial diagram shown in the previous slide which is on the left and on the right I have F 1 parallel F 2. So, P 1 parallel P 2 and F 1 parallel F 2 is on the right side and these two are the individual composite models for the function that is the functions and the resources. Now, when the resources and the function work together the composition of F 1 and F 2 gives the composite system model the system model consists of the processes that will be executing and also the resources which will be consumed by the processes.

Now, the behaviour of the resources when composed with the behaviour of the process of the processes gives the composite system model P f and; that means, given by P 1 parallel P 2 parallel F 1 and F 2 and here we see that sigma of P 1 parallel P 2 has the has all the states has all the possible has all the six states that we have defined and F 1 parallel F 2 has all the six states that we have defined. Therefore, the composite automata must synchronize on all events. So, only the first line of the transition functions. So, again the transition function del q 1, q 2 comma sigma goes to del 1, q 1 comma sigma gamma del q 2 comma sigma this is the only applicable one because all events are common. All events are common in both the automata and hence they must synchronize on all events and because this is going to be a big automata we again show a partial diagram.

So, let us take the initial state of the composite behaviour. So, what is the composite behaviour? The philosophers are in state are in thinking state and the resources are available. So, T 1 T 1 comma T 2 the philosophers are thinking and A 1 comma A 2 the resources are available. Let us say in this state you take the you take that event the event f 1,1 occurs. So, if f 1, 1 occurs then in the ca in the ca in P 1 parallel P 2 I must go to I 1 comma T 2, ok. So, I will go to I 1 comma T 2 and it is if f 1,1 is also defined in if f 1,1 is also defined in F 1 parallel F 2 at a 1 comma a 2 at the available state and. So, F 1 parallel F 2 moves to U 1 comma A 2, and by using the transition function we move to the state I 1, 1, T 2 U 1 A 2. So, I move to I 1, 1 comma T 2 of P 1 parallel P 2 and U 1 comma A 2 of F 1 parallel F 2. So, in the composite automatic I move to I 1, 1 comma T 2 comma U 1 comma A 2.

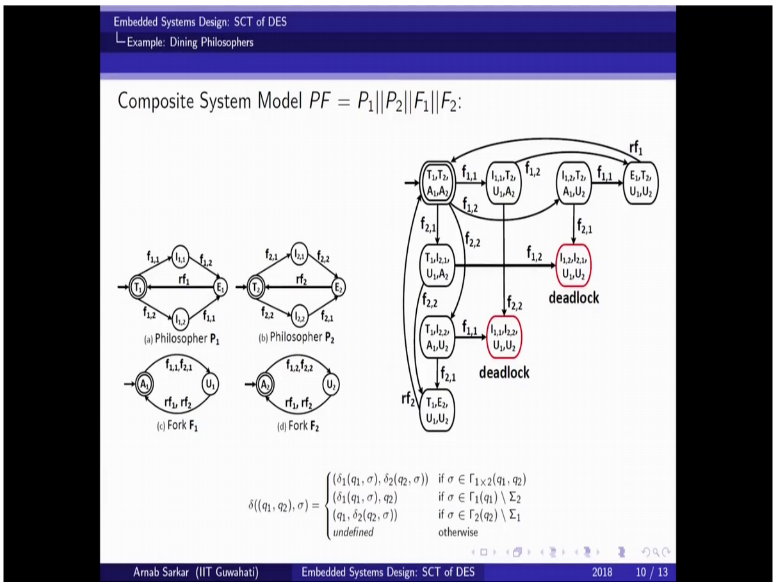
And, then let us say at the state I 1, 1 comma T 2 U 1 comma A 2 I the event f 2, 2 occurs. What is event f 2, 2? Philosopher 2 picks up fork 2. So, remember f 1,1 occurred previously meaning that philosopher 1 had picked up fork 1 and then we moved to I 1,1 T 2 U 1 A 2 and now, philosopher 2 has picked up fork 2. So, I moved to I 1,1 comma I 2,2 and U 1 comma U 2. So, what is the event? Let us say if you look at P 1 parallel P 2 and I am at state I 1,1 comma T 2 and the event f 2,2 occurs we see that we move to the state I 1,1 comma I 2,2 and let us say we are at T 1 comma A 2 and the event f 2,2 occurs where do I and the state f 2,2 occurs and I go to the state U 1 comma U 2 that is both the forks are used.

So, now, I have reached a state where both the philosophers are in that intermediate state, none of them has reached the eating state and both the forks are in the nude state. Why? Because philosopher 1 has picked up fork 1 and philosopher 2 has picked a fork 2, none of the philosophers have been able to pick both the forks. Now, at this if at this state we see that there are no common events meaning that at I 1, 1 comma I 2, 2 at I 1,1 comma I 2,2 none of the events that are active at U 1 comma U 2 are defined. So, what are the events that are activate at U 1 comma U 2 of f 1,1 parallel f 2 2? Only rf 1 comma rf 2. So, rf 1 and rf 2 are defined as active events at U 1 comma U 2 of F 1 parallel F 2 and these two events are not defined on I 1,1 comma I 2,2.

Similarly, so, the automata cannot move from the composite automata cannot move from the state I 1,1 comma I 2,2 comma U 1 comma U 2 because rf rf rf 1 comma rf 2 is not defined at this composite state. Similarly, f 1 f 2 or if f 1,1, f f 1,2, f 2,1, f 2,2 all these the picking up of forks these are not defined in U 1 comma U 2 ok, but is defined in I 1,1 comma I 2,2. So, on these events you cannot move. So, there is no way on which you can move in the composite automata from the state I 1,1 comma I 2,2 comma U 1 comma U 2. There are no defined active events on which you can move and therefore, but and the final and the final mark state of this automata has still not been reached because the marked state is what T 1 comma T 2 comma A 1 comma A 2, T 1 comma T 2 comma A 1 comma A 2 and that state has not been reached.

Hence we have reached a deadlock in which there is no way in which I can move to the marked state. So, therefore, this concurrent uncontrolled behaviour of the system contains deadlock states which must be avoided by the controller. And it is absolutely possible that you can reach these deadlock states if not controlled appropriately and you will never be able to reach the marked state and you will remain deadlocked with none of the processes being able to move in the embedded system.

(Refer Slide Time: 61:10)



Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

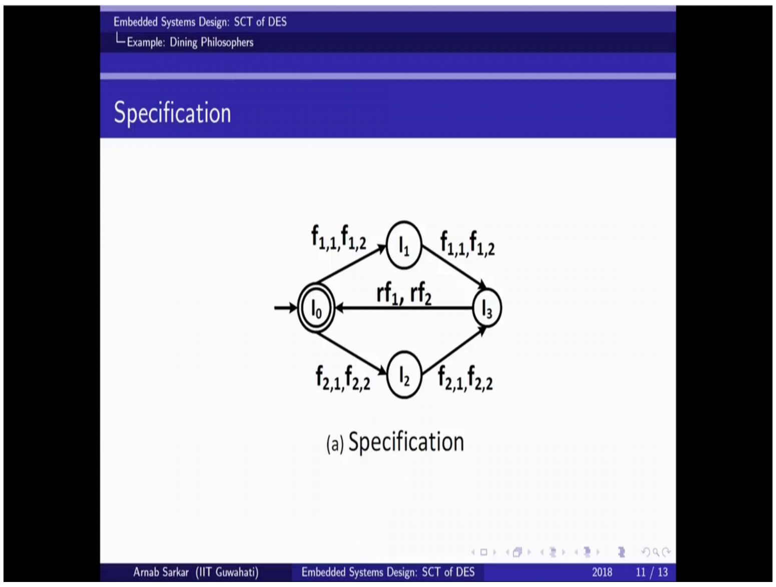
Composite System Model $PF = P_1||P_2||F_1||F_2$:

$$\delta((q_1,q_2),\sigma) = \begin{cases} (\delta_1(q_1,\sigma),\delta_2(q_2,\sigma)) & \text{if } \sigma \in \Gamma_{1\times 2}(q_1,q_2) \\ (\delta_1(q_1,\sigma),q_2) & \text{if } \sigma \in \Gamma_1(q_1) \setminus \Sigma_2 \\ (q_1,\delta_2(q_2,\sigma)) & \text{if } \sigma \in \Gamma_2(q_2) \setminus \Sigma_1 \\ \textit{undefined} & \text{otherwise} \end{cases}$$

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 10 / 13

So, if we draw this draw if you do this composition completely we get these composite automata on the right. The very partial automata that we showed in the previous tab that we showed in the previous slide here on the right side c, c part on the right side; this path or these two transitions can be seen how here. So, from T 1,1 in if you see the right automata if you see the right composite automata from T 1, T 2 A 1 A 2 on F 1 I go to I 1,1 T 2 U 1 A 2 and then I take F 2,2 to come to the deadlock state. So, this part is present in here. So this represents the this figure this model on the right that you have represents the composite behaviour over using all the resources and processes that I have in the embedded system.

(Refer Slide Time: 62:07)


Embedded Systems Design: SCT of DES
└Example: Dining Philosophers

Specification

$$\begin{tikzcd} & l_1 \arrow[dr, "{f_{1,1},f_{1,2}}"] & \\ \arrow[r] & l_0 \arrow[ur, "{f_{1,1},f_{1,2}}"] \arrow[dr, "{f_{2,1},f_{2,2}}"'] & & l_3 \arrow[ll, "{rf_1, rf_2}"'] \\ & l_2 \arrow[ur, "{f_{2,1},f_{2,2}}"'] & \end{tikzcd}$$

(a) Specification

Arnab Sarkar (IIT Guwahati) Embedded Systems Design: SCT of DES 2018 11 / 13

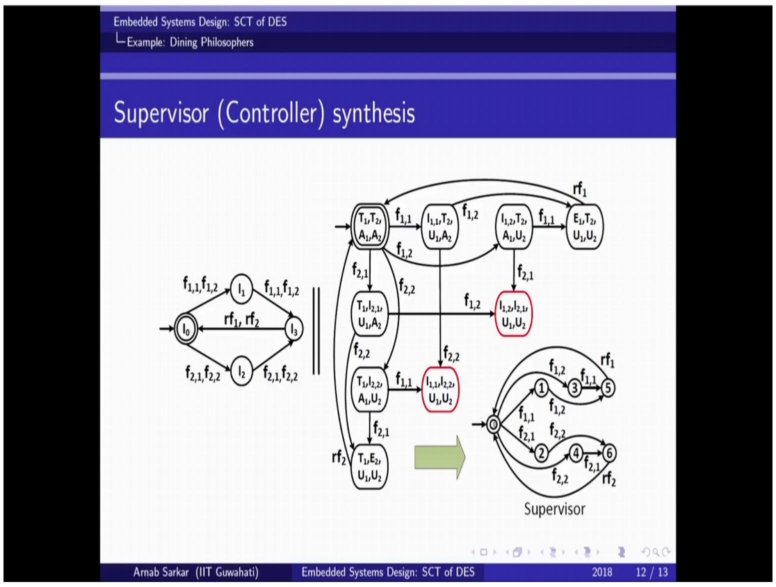
Now, given this I have to be able to control these automata. The, I have to control this concurrent unrestricted behaviour. So, how do I control this concurrent unrestricted behaviour? By designing a specification that we want so, what do we want? We want we want that if fork 1 is picked up by philosopher 1 only he should be allowed to pick fork 2 or if fork 2 is picked up by philosopher 1 only he should be allowed to pick fork 1. The other way for the philosopher 2 if philosopher 2 picks 1 of the forks only he should be allowed to pick the other fork and this is say enforced by this specification automata here which we need to define our model.

So, in the specification or in the legal specification that we want in the initially I am in state I 0 at state I 0 either eh philosopher 1 if I take the top transition the transition on the top what happens, either fork 1 or fork 2 is picked up by philosopher 1 and I go to intermediate state I 1. So, intermediate state I 1 tells me that philosopher 1 has picked up either fork 1 or fork 2. At I 1 therefore, what are the valid events that I should allow? I should allow philosopher 1 only to pick the other fork.

So, the valid events at I 1 are f 1, 1 or f, 2; that means, philosopher 1 should be allowed to pick the other fork. So, I if I have reached I 1 from I 0 1 f 1,1 then I will allow the I will allow f 1,2 to occur at eh I 1, so that I can go to I 3. So, if I have used up f 1, 1 f 1, 1 is no more possible at I 1 because fork 1 has already been picked. So, f 1, 1 is not possible. So, what is possible and allowable only is f 1, 2. So, I should disallow f 2, 2 at I 1.

Similarly, if you look at their transition on the bottom from I 0 philosopher 2 can pick either fork 1 or fork 2 and move to I 2 and the other fork philosopher 2 can only be allowed at I 2, philosopher 2 is only allowed to pick the other fork and move to I 3. At I 3 either philosopher 1 or philosopher 2 relinquishes both the forks and moves back to I 0 which is the marked state as well. So, it is the initial as well as the marked state. Now, given this legal specification what we do is then compose what we do is then compose parallel composition similar parallel composition of the system of the specification and unrestricted behaviour.

(Refer Slide Time: 65:22)



Supervisor (Controller) synthesis

So, we compose the specification which is what the system should do legally and unrestricted behaviour and when we do that we obtain a supervisor or the controller which is shown at the bottom at the right of the arrow. So, at the right of the green arrow shows what is the supervisor that we get through the composition. Let us see one or two states how this transition how this transition system or the supervisor of the supervisor is obtained.

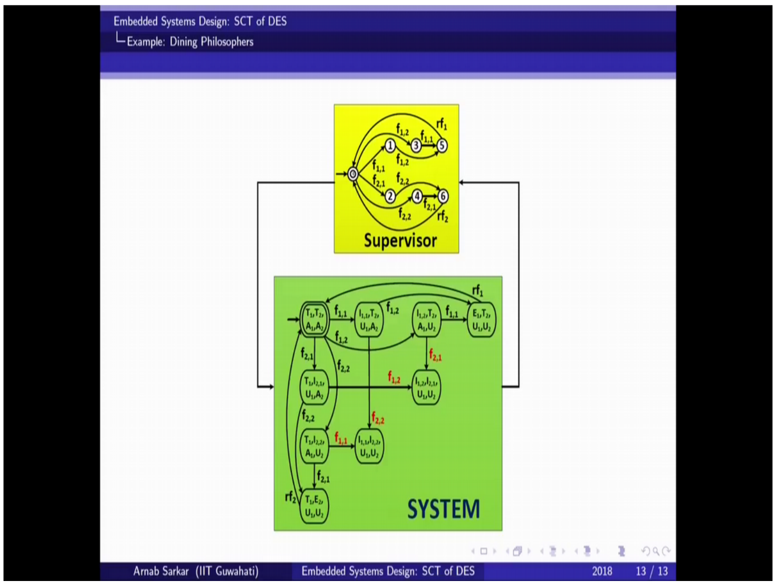
So, initially I will be in T 1, T 2, A 1, A 2 of the unrestricted composite behaviour and I will be in state I 0 of the specification. At that state let us say I allow f 1, 1 the event f 1,1 occurs. So, where will I go? I will go to state I 1 of the specification and to the state I 1,1, T 2, U 1 comma A 2 of U 1 comma A 2 of the composite unrestricted system model. So, at this system model we see that if I am in state I 1, 1, f 2, 2 is not defined. However, f 2, 2 is defined in the automata. So, the transition system because all events both the specific the events in the specification and the events. So, we have a total of six events and all the six events are there in the specification all the six events are there in the unrestricted behaviour. So, with during composition if the system must synchronize on all events, ok. The composite system must synchronize on all events.

So, therefore, only the first line again going back of the transition system del is applicable of the transition function del is applicable, ok. So, because f 2, 2 is not allowed at I 1 so, in the composite behaviour in the composite behaviour in the composite controlled behaviour the event f 2, 2 will be disallowed. So, there will be no event the in the composite behaviour I cannot go from I 1 comma I 1,1 comma T 2 comma U 1 comma A 2 which is the composite state and go to a state such as I 1 comma I 1,1 comma I 2,2 comma U 1 comma U 2 such a transition to such a transition is not allowed.

Now, if you see the composite automata the supervisor figure here. So, initially I am in state 0 I go to f 1, 1 and I move to the state 1. At state 1 at state 1 of the supervisor or the controller I see that the event f 2, 2 as we discussed has been disallowed I have not allowed f 2, 2 what have I only allowed the only possible allowing which if you go back to the specification at I 1, 1 the two events that are possible are f 1, 1 and f 1,2 and because f 1,1 has only has already been taken the only event that is possible is f 1,2. So, at 1 I allow f 1, 2 to happen and I go to 5.

Therefore, the specification eh when I when I compose or parallel compose the specification with the uncontrolled behaviour I get the supervisor which disallows transition to the deadlock states of the deadlock states of the composite behaviour, ok. So, I nm the event the controller will only allow those events which do not lead to deadlock states the controller or the supervisor is such that it allows only those events which disallows transition to the deadlock states.

(Refer Slide Time: 69:31)


Embedded Systems Design: SCT of DES
Example: Dining Philosophers
Supervisor
SYSTEM
Arnab Sarkar (IIT Guwahati)
Embedded Systems Design: SCT of DES
2018
13 / 13

Now, after this finally, we how does it occur how does the entire system function? I have the unrestricted system which is the system behaviour the physical system. Now, let us go back and think of the cyber physical system or the con control system that we had defined the cyber physical system or the control system that we had defined in lecture 1. We had a physical system which is the system in green shown here we had a controller which controls the physical system which is shown in yellow over here that is the supervisor I sense what is the current state of the physical system through a sensor that is fed to the controller and the controller generates appropriate actuation actions.

Now, suppose thus the initial system and at the initial state of the system f 1,1 is taken. So, I know that the sensor tells me that I am I was in the system I am in state 1 and I have taken the event and the event f 1,1 has occurred. So, the supervisor sends from the sensor the input that I am in state and event and event f 1,1 has occurred this goes to the supervisor and the supervisor then says that what are you allowed to do?

You are allowed to take only the supervisor therefore, goes in this in the in similarly supervisor accordingly goes from the initial state on f 1,1 goes to 1 and then the supervisor says that you are only allowed to take f 1,2 and you are not allowed to take f 2,2. So, the actuation that it tells the actuator that please tell the physical system to only allow f 1,2 and not allow f 2,2 because f 2,2 will led will lead to a deadlock state and thereby we have obtained the controller which controls the physical system.

So, in this lecture, we what we have we studied? We have studied how to formally model the behaviour of the entire system and we studied how to obtain a controller. Now, there are tools such as supreme cart, TTCT, there are automated tools where you can define these individual models and there are automated. So, these formal transition methodologies on which these transition functions will be automatically applied, the tools will automatically apply this transition function, compose them and we will have a correct by construction supervisor controller depending on the behaviour.

So, if the behaviour of the system is it has been modelled correctly, if the specification has been modelled correctly then we have a current correct by construction procedure by which we will obtain a supervisor or controller for the system that can be applied on the on the physical system to control it appropriately.

With this we come to the end of this lecture.