INDIAN INSTITUTE OF TECHNOLOGY
KHARAGPUR

NPTEL
ONLINE CERTIFICATION COURSE

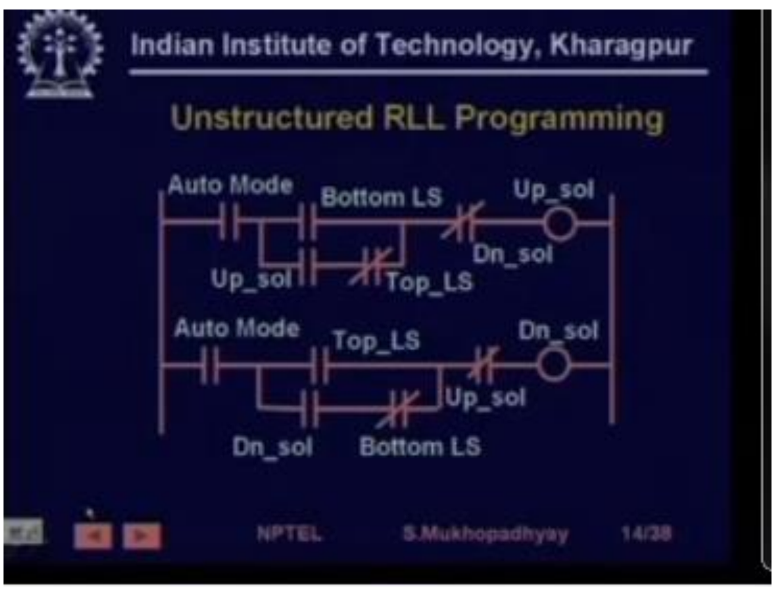
On Industrial Automation and Control

By,
Prof. S. Mukhopadhyay
Department of Electrical Engineering
IIT Kharagpur

Topic lecture – 29
A Structured Design Approach
To Sequence Control (Contd.)

So this is a typical example of unstructured programming.

(Refer Slide Time: 00:25)



The kind of things that we did before this lesson.

(Refer Slide Time: 00:30)

Now having said these things so this is in very brief this is a structured design approach to RLL programming now we must mention since we are coming to the end of this programming part we must mention that there are you know certain standards of programming languages which have come for example IEC 1131, IEC stands for the international electro technical commission.

(Refer Slide Time: 01:00)

And the 1131 is a number so this is an international standard for PLC programming and it classifies languages into two types one is grown a graphical languages.

(Refer Slide Time: 01:09)

Things like you know functional block diagrams or ladder diagrams these are graphical languages on the other hand there may be some several text based languages for example structured text or instruction list these are some of the kinds of programming paradigms I already told that that although we are learning about the RLL there are several other programming paradigms which are also supported by various manufacturers. And these are typical examples of them.

(Refer Slide Time: 01:33)

So for now we have understood that this state based probe design is very useful so therefore to be able to capture this state transition logic a separate language has been proposed separate methods of programming advanced programming is have been proposed and we are going to take a brief look at that so some of the merits of this kind of you know advanced programming which takes this kind of abstract state transition logic firstly because they are they have open standards in the sense that anybody can write a program which can be used by somebody else.

(Refer Slide Time: 02:20)

Then structuring of programs in terms of module helps in program development and program maintenance very important program ablation you are you want to add a new functionality you will find that it is very simple just change the state diagram and then find out that now in the new state diagram if you have three extra states simply you have to add those corresponding rungs if you have some extra transitions you have to add those rungs and if you have to redirect some transitions to now new states then you have to code that logic that is that transition should now come in that state logic.

So you absolutely know that which are the places where you have to make change these are the very standard benefits of structured programming.

(Refer Slide Time: 03:07)

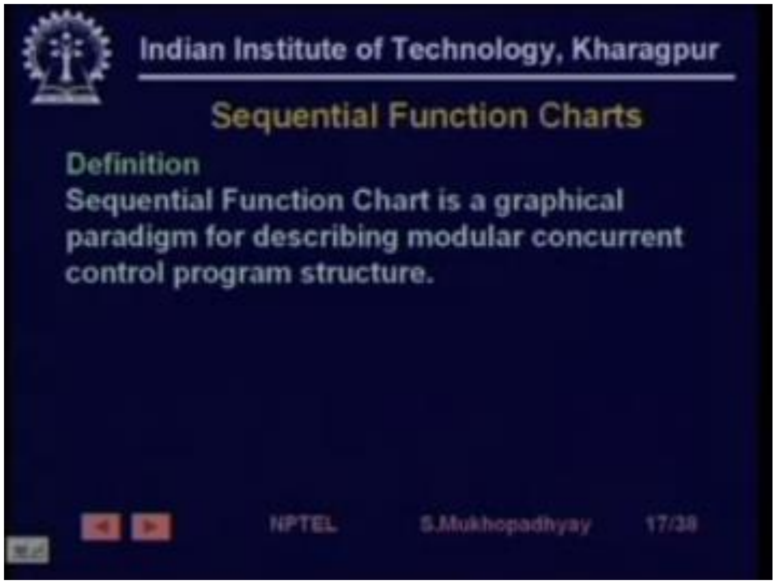
Then it is not necessary that all the times every rung has to be computed in fact only a few rungs are actually active and the other ranks are inactive so there is no point you can save lot of computational time by skipping those so this needs to be done that will improve computational performance.

(Refer Slide Time: 03:29)

And it supports concurrency that is very important that it supports concurrency because it very often happens that there are there are many things several things which will be taking place together and they which are which are best modeled as concurrent and especially now that we have you know this so called multitasking operating systems or executing running on microprocessor.

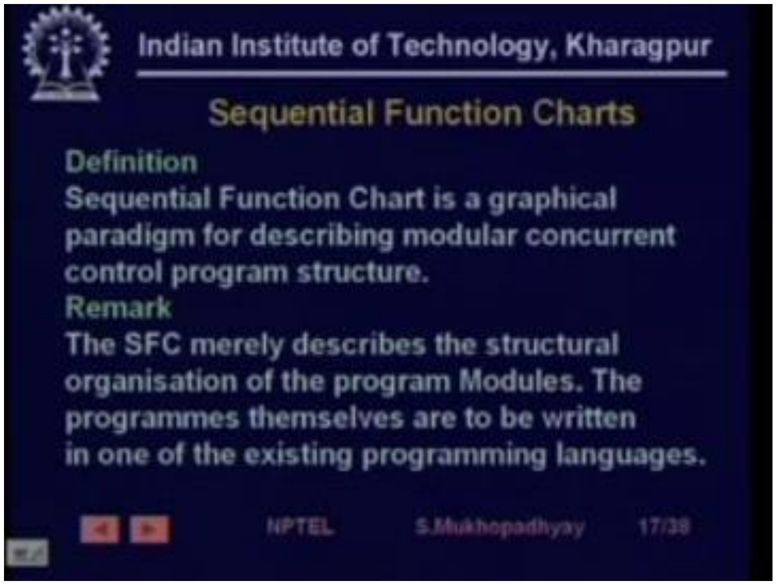
So there is there is no reason why this cannot be done I mean we have the technology to enable it so therefore we must use it so therefore we have a formalism which is we standard and which are called sequential function charts so a sequential function chart is actually a graphical paradigm for describing.

(Refer Slide Time: 04:17)



Modular concurrent control program structure so remember that it is basically used to describe structure and not the program itself so which part so as if you have organized your program into a number of well written functions which are modularly arranged so you just using the sequential function chart you are just describing that which of these functions will be executed when and under what condition.

(Refer Slide Time: 04:48)



So you have this remark which says that the SFC merely describes the structural organization of the program modules while the probe the actual program statements in the modules still have to be written probably using existing PLC programming languages something like you know are either RLL or instruction list or whatever.

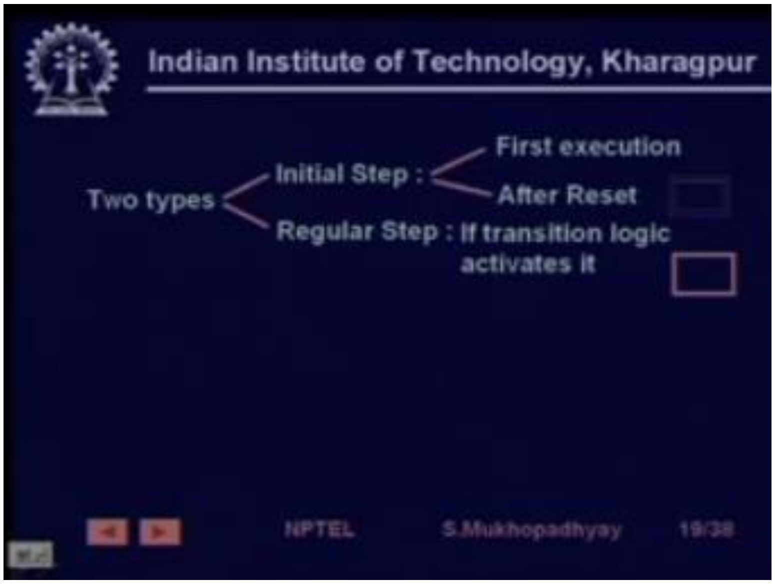
(Refer Slide Time: 05:13)

So if let us look at quick look at the basic SFC constructs they are basically state machine like constructs and contain some constructs which are which simple programming constructs with which we are familiar are.

(Refer Slide Time: 05:26)

So you have basically just like states and transitions here you have steps and transitions so each step is actually a control program module which may be programmed in RL or any other language so as I said it is a function.

(Refer Slide Time: 05:39)



Similarly this steps can be of two types number one initial step I had remarked the initial step and the initial step execution can be of two types one is when the first time we are executing after power-on and second is if a program reset actually sequential function charts have some standard instructions which will reset which are assumed to be reset assume to reset the program so after reset also you can execute a particular type of step called the initial step typically used for initializing variables and states and intermediate variables.

Otherwise you have regular steps we which one of them and active at any time actually depends on the transition logic as we have seen.

(Refer Slide Time: 06:25)

So were a step becomes inactive its state is initialized and only active steps are evaluated during the scan that saves time.

(Refer Slide Time: 06:36)

Now we have transitions so in transition each transition is also a control program module which evaluates based on available signals operator inputs etcetera which transition conditions are getting enabled so once the transition variable evaluated true then the steps following it are activated and we say steps because we in this formalism typically we have seen that one transition can go to only one step provided you do not have corn currency but since SFC is support concurrency.

So it may happen that after a one transition to two concurrent you know threads of execution will start so that is why we have said steps following inter activated and those preceding it are deactivated.

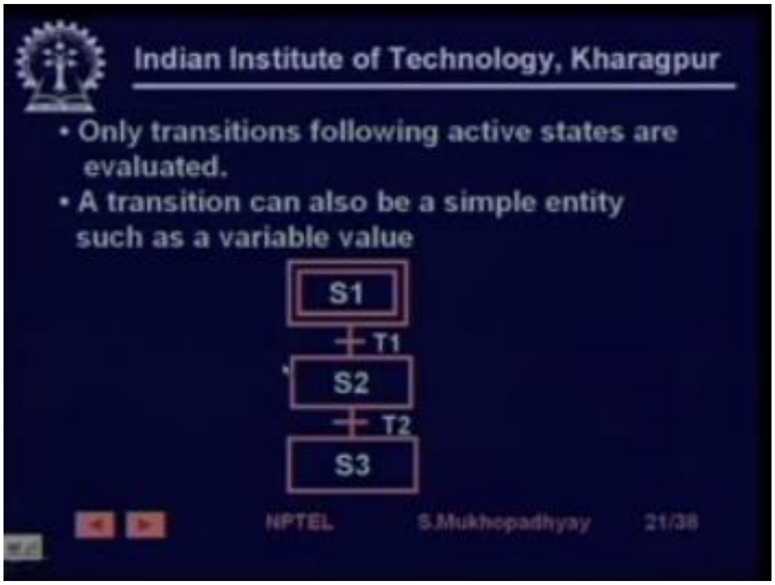
(Refer Slide Time: 07:23)

Only transitions following active state so again when you are in an inner state already which is active only certain transitions can occur so therefore only those transitions need to evaluated so we only those are evaluated.

(Refer Slide Time: 07:39)

A transition can also be a, it can either be a program itself having embodying very complicated logic or it can be a simple variable like in our example which is simple enough.

(Refer Slide Time: 07:52)



So for example in this case it says that S1 is an is actually an initial step and then if transition while it is in while S1 is active T1 is active and if the conditions 41 are fulfilled their will come to S2 if they if I evaluate S2 if T2 will become also active and if T2occurs then it will goes to S3.

(Refer Slide Time: 08:17)

So you see that if you see the activations in scan one it may so happen that S1 and T1are active while in scan to it may happen that T1 has fired once T1 has fired that is actually 2 taken place because T1 is active so it has come to S2 and thenT2 is active so if T2 is active it is continuously evaluated it may happen that in scan three T2 is evaluated to be false so these two continue to be active.

Why let scan for it may happen that T2 is now actually become true so therefore S3 has become active and then some a transition out going from it will become active so this is in this way certain parts of the program at a time becomes active as the system moves through the sequential function chart.

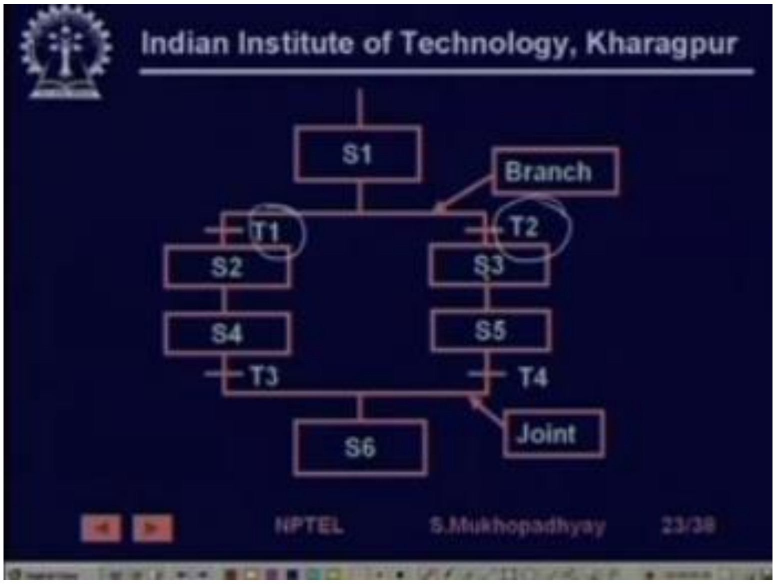
(Refer Slide Time: 09:15)

For example there are you can you can describe several kinds of you know programming constructs for example.

(Refer Slide Time: 09:22)

This is an, I am sorry.

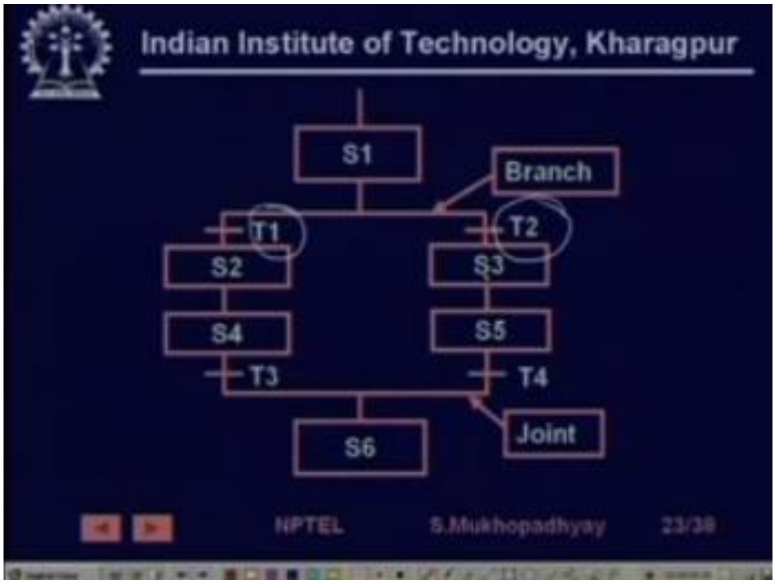
(Refer Slide Time: 09:26)



So for example this is the case where the system if at S1 at while the system is at S1 T1and T2 are active so any one of them can become true and if T1 is true S2 the next active set will be S2 if T2 is to true the next active step will be S3while it could conceivably also happen that S2 a T1 and T2 in a in a given scan T1 and T2 both become active so in that case you have to dissolve because both transition cannot take place simultaneously.

So you have to say that so there is there is a convention that the leftmost transition becomes takes place actually takes place.

(Refer Slide Time: 10:11)



So if T1 and T2 will both become active at the same time true at the same time then T1is assumed to have taken place.

(Refer Slide Time: 10:23)

So that was a selective or alternative branch that is the system can either flow through this branch or flow through this branch but not both of them simultaneously.

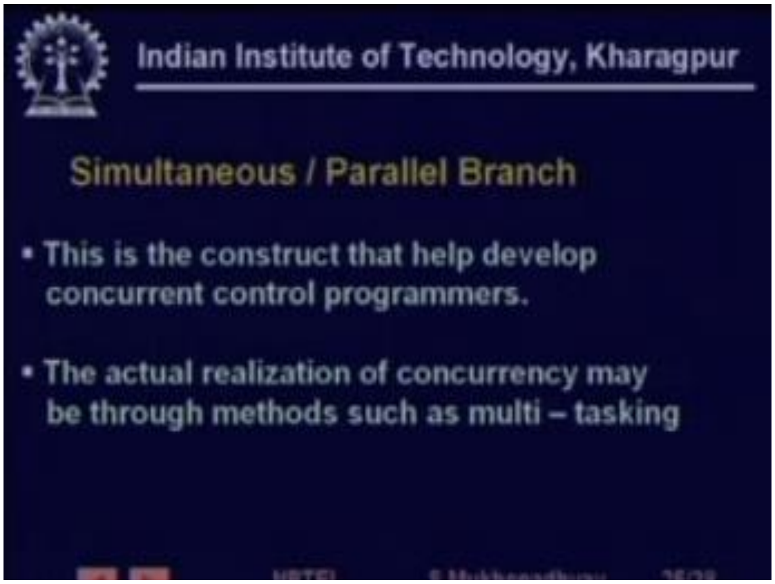
(Refer Slide Time: 10:35)

So as I say that if S1 is active T1 is true and then S2 is becomes active if T1 is false and T2 is true then S3 becomes active and then the moment S3 becomes active S1 becomes inactive similarly left to right priority as I said and only one branch can be active at a time if S4 is active and TC becomes true s6 becomes active and S4 becomes inactive same thing.

(Refer Slide Time: 11:01)

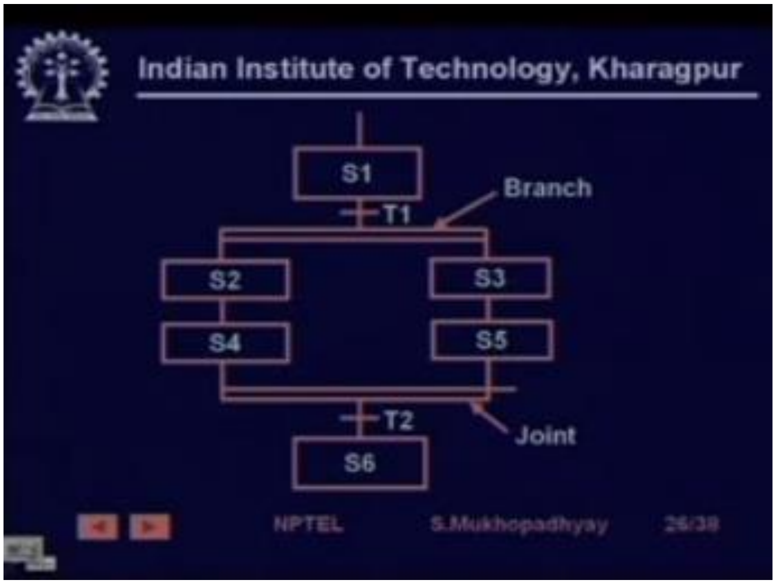
Similarly just like selective or alternative branches you can also have simultaneous or parallel branches so this is the new construct which is which we which is not possible to beacon you implemented in a normal state machine.

(Refer Slide Time: 11:16)



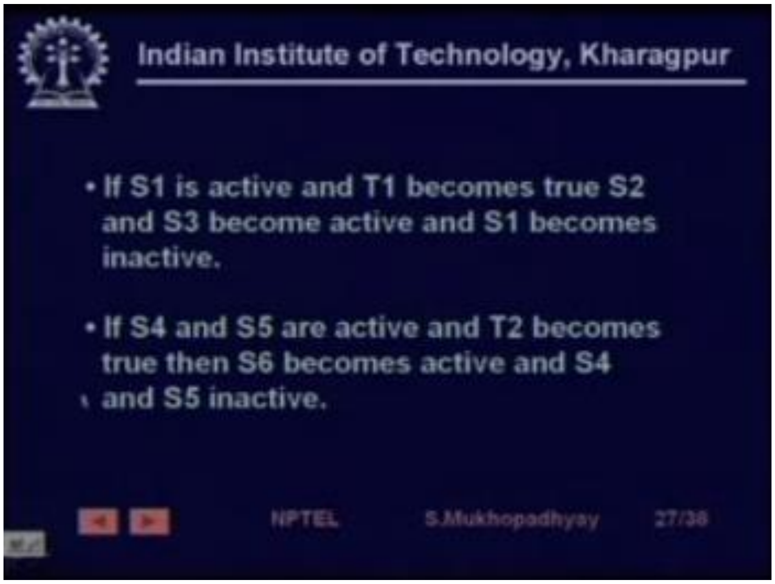
So this is a construct that helps concurrent control programs the actual realization how you will run context and concurrent executions the can be made using various you know operating system methodology is such as multitasking.

(Refer Slide Time: 11:33)



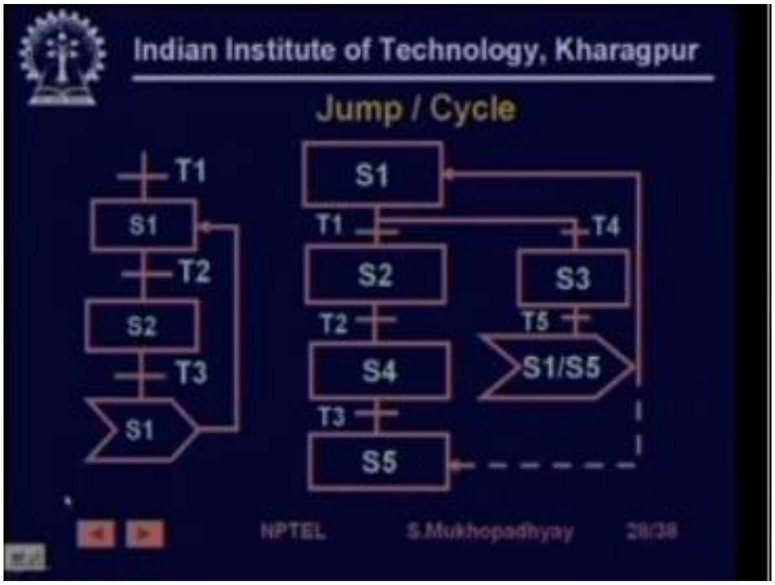
So here you have a parallel branch so when you see that address one if T1 occurs then both these states become active together and then at some point of time S4 becomes active but T2cannot take place unless this branch also comes to S5 so only when S4 and S5will become active at that time T2 will become active and it will be evaluated so if it evaluates to true then it will come to S6 and then S4 and S5 will become inactive.

(Refer Slide Time: 12:09)



So if that is there is the same thing is written here if S1 is active and T1 becomes true S2 and S3 become active and S1 becomes inactive if S4 and S5 are active and T2 becomes true then s6 becomes active and S4 and S5 inactive okay.

(Refer Slide Time: 12:25)



Similarly you can have you know among program control statements you can have jumps so for example you have a jump were after S1 S2 after T3it will go back to S1 similarly you can have a jump inside the loop you can, I mean jumping here you can either come to S5 or you can come to S1 so basically you can execute this kind of program control statements using SFC.

(Refer Slide Time: 12:53)

So a jump cycle jump occurs after transitions and jump into or out of parallel sequence is illegal because you cannot jump out of one sequence when you are in parallel you must enter them together in any parallel sequence while you are executing the sequence you maybe in different states at different arms but you must enter them together and you must leave them together.

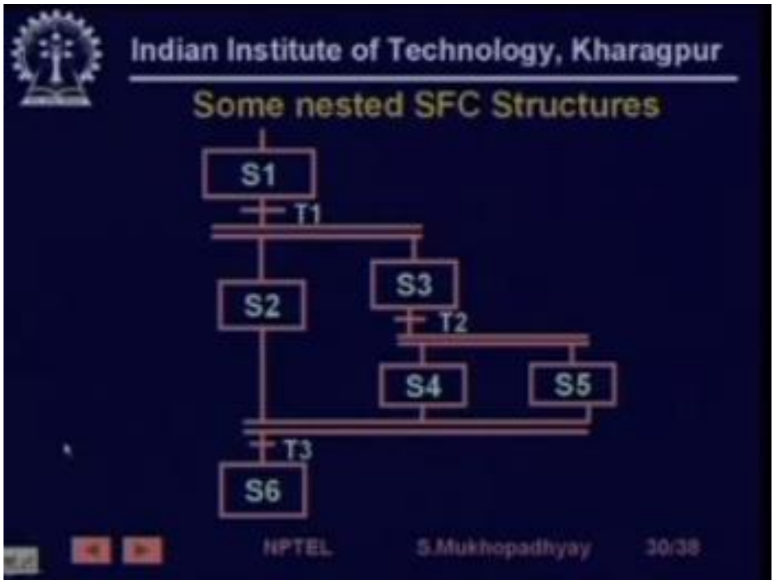
(Refer Slide Time: 13:15)

Similarly last you may have termination of a program so step without a following without a following transition if you have a step out of which there is no transition there is a dead state.

(Refer Slide Time: 13:28)

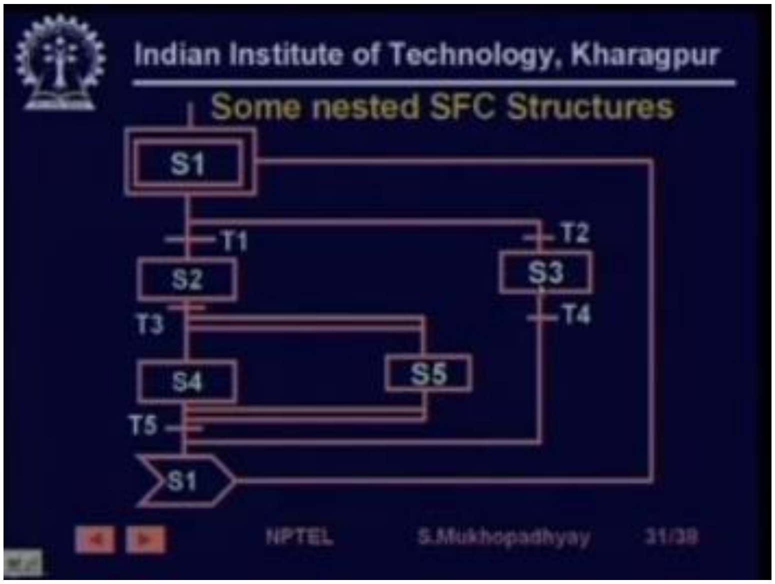
And it once it is activated it can only be deactivated by an special instruction for SFC reset.

(Refer Slide Time: 13:33)



You can nest programming you can nest a concurrent do within a concurrent loop so here T1 authors S2 and S3 then from S3 if T2occurs then simultaneously S2 S4 and S5can become active and if all of them are active thenT3 will take place and then S6.

(Refer Slide Time: 13:54)



Similarly you see here what is happening this exactly the same thing is happening that you hear you have a selective branch so you can have you can you can either go into S2 through T1 or go into S3 through T2 that is a selective branch while you are in a selective branch you can execute a parallel branch and then if when S4 and S5 are both active at that time if T2evaluates to true then you will come to a jump which will again take you to S1. So such control flows you can specify.

(Refer Slide Time: 14:38)

So we have we have come to the end of the lesson in this lesson we have what have we learnt we have learnt the broad steps in the in the sequence control design.

(Refer Slide Time: 14:53)

And these are very important that first of all identify the inputs or outputs then very critically study the system behavior look at the requirements for manual control operator safety fault etcetera and then try to formalize this description so first maybe write the operation in crisp steps in English and then try to convert this description of steps into a some sort of formalism and we have in this lecture we have seen the formalism of sequential state machines. Which can be programmed using a sequential function chart.

(Refer Slide Time: 15:45)

So this modeling we have seen that how to model a particular application we took a very simple industrial application like a stamping process and built it state machine and then we have shown that how given the state machine how very mechanically you can arrive at how you can you can actually structure your program and then write the transition logic and the state logic and the output logic. And we have seen that this is very much expected to lead to you know error-free programs.

(Refer Slide Time: 16:17)

And finally we actually found a syntax I mean some language graphical language which can actually capture this flow of logic in terms of states sets and transitions so this program control or the program flow description strategy using sequential function charts we have seen so now before ending I think it is nice to look at some problems so for your examples you can try.

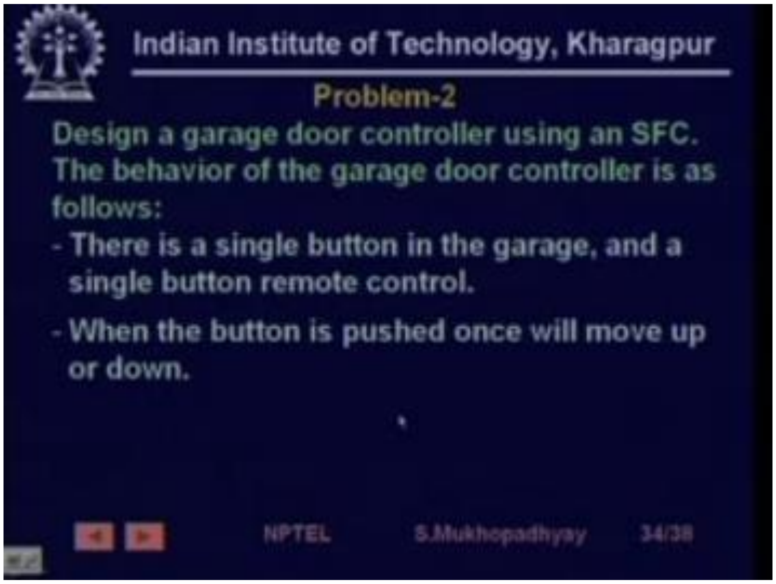
(Refer Slide Time: 16:56)

You can try to create let us look at exercise problem number one so the problem number one says create an RLL program for traffic light control so what happens that the lights should have crosswalk buttons for both direction of traffic lights okay a normal light sequence for both directions will be green a normal light sequence for both directions will be green 16 seconds and yellow four seconds.

If the crosswalk button has been pushed then a walk light will be on for 10seconds if the crosswalk button has been pushed then what light so will be on for10 seconds and the green light will be extended to 24 seconds right so this is see this much of description is only given now while so what is the task the first task would be to develop a state machine description and probably write it in terms of write it in terms of an SFC.

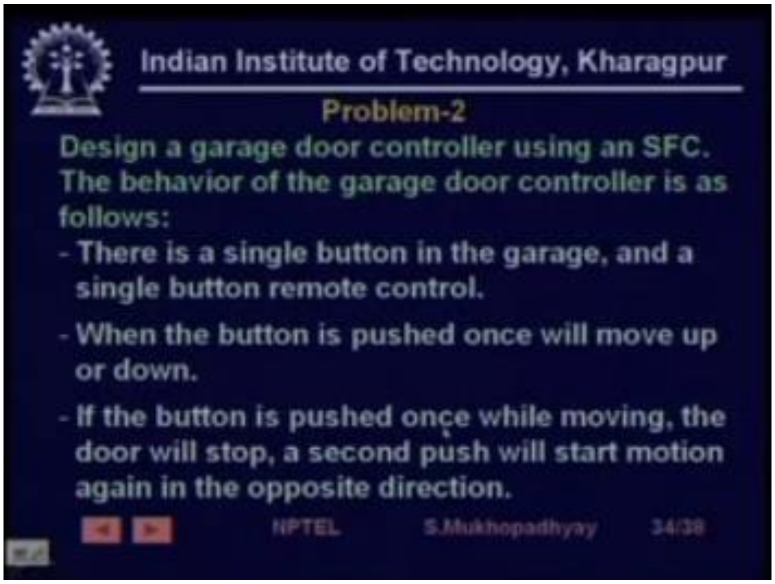
And then while going to do this is SFC we will actually encounter we can encounter several problems for example we may find that things are not some things are not straight for example let us look at the let us look at the SFC of this problem.

(Refer Slide Time: 18:30)



So we will for the time being we'll skip problem number two or okay let us look at problem number two next and then we will go to the recipes so the next problem says design a garage door controller using an SFC so the behavior of the garage door controller is like this there is a single button in the garage and a single button remote control when the button is pushed the door will pushed once when the when the button will be pushed once then the then the door will move up or down.

(Refer Slide Time: 19:17)



So you push it once it will move up or down if the button is pushed once while moving then the door will stop and a second push will start motion again in the opposite direction so you have a single button which you are going to push so you so you press it once it might go up press it again it will stop press it again it will reverse so you have only one single button so you see that this is purely sequential behavior.

So unless you have a concept of state you can never you are actually pressing the same button but sometimes it is stopping sometimes is moving up sometimes moving down and this kind of logic it is not possible to model using only input output then you have to bring in the concept of memory and state as you might have noticed when you have designed digital logic circuits so you have sequential when you have whenever you have memory you have sequential logic.

(Refer Slide Time: 20:09)

So let us look at then there are top bottom limit switches to stop the motion of the door obviously you have pressed a button it is going up it has to stop somewhere so you have to have limit switches.

(Refer Slide Time: 20:16)

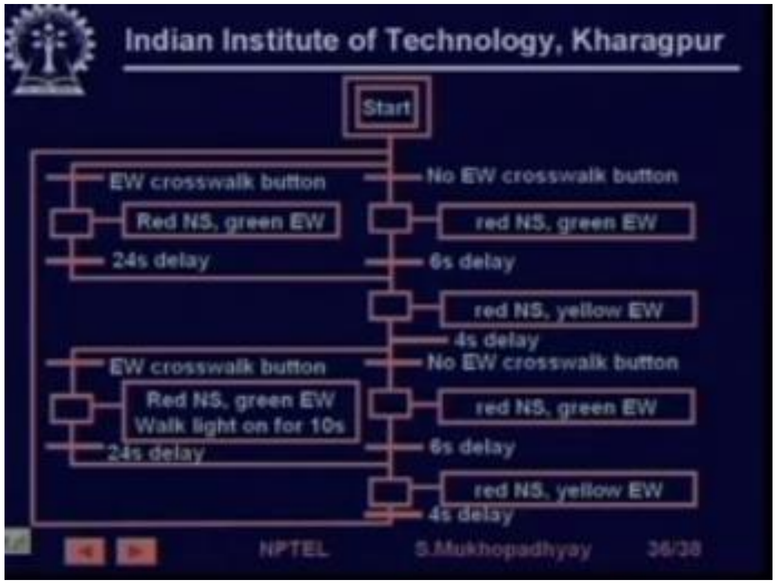
There is a light beam across the bottom of the door if the beam is cut while the door is closing then the then the door will stop and reverse so if you have while the door is closing suppose if you if you put something below it then that is you are trying to ensure that the door car actually closed because otherwise what will happen is that the door will get stuck if you have kept anything there suppose the back of your car is actually sticking out.

Then the door will go and hit your the door will go and hit your car, so the door will close fully only if there is a light beam which is not interrupted which means that the less clearance otherwise that if the light we will interrupt at any time immediately the door will stop and reverse so you think that there is something it cannot close it.

(Refer Slide Time: 21:14)

There is a garage light that will be on for five minutes after the door opens or closes so you have a garage light which can be open for five minutes after the door opens or closes so you have to use a timer here so sorry any time you do an operation the light is going to go on and stay five minutes okay.

(Refer Slide Time: 21:41)



So we have the two let us first look at the next one which is simpler.

(Refer Slide Time: 21:43)

So here you have the garage door so let us see how they have modeled it so you have in this case so if button is pressed and so you have step one going to step two going to step 3 this is the closed door so if let us see let us go back.

(Refer Slide Time: 22:10)

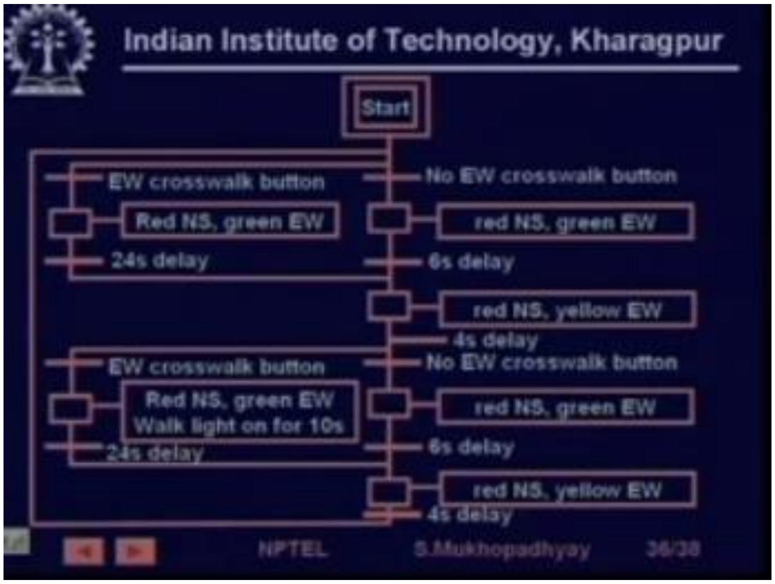
(Refer Slide Time: 22:12)

So there is a single button in the garage and a single button remote control so you can have either a button pressed in the garage or you can have a button press in button pressed from the remote.

(Refer Slide Time: 22:27)

Right.

(Refer Slide Time: 22:28)



So there are.

(Refer Slide Time: 22:29)

Two kinds of buttons so you have either a button press from the garage or from the remote then we go to step 3 and start closing the door this is the this is the output on the other hand is button has been so if from there if either a button has been pressed either from local or from remote or if so if we if we press the button what will happen then the door will stop on the other hand if the if the limit switch is made what will happen the door will stop.

So if either a button has been pressed or the bottom limit switch is reached then immediately the door will stop or if the light beam is interrupted then it will not only stop it will actually reverse right so immediately it is going to reverse here what happens is that if you have pressed a button and to stop it then if you press it again then it will go to reverse so this is a you see we have we have captured the behavior of the garage door in the in this form using SFC so you can do you can do a similar thing also for the traffic light that let that be our necks and exercise so that brings us to the end of this lesson, thank you very much and see you again for the next lesson, bye, bye.