Industrial Automation & Control

Prof. S. Mukhopadhyay
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

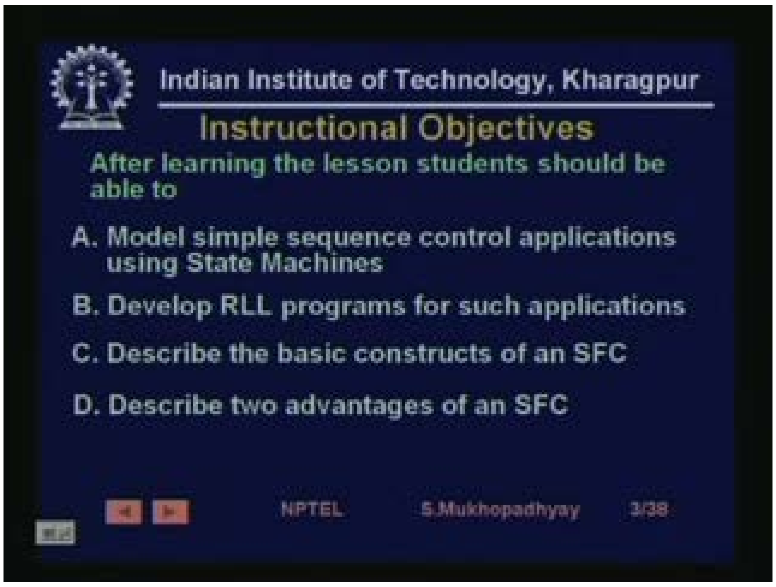
Lecture - 21

A Structured Design Approach to Sequence Control

Welcome to lesson 21 of the course Industrial Automation and Control, in this lesson we are going to learn A Structured Design Approach to Sequence Control. So far we have mainly seen the programming constructs have seen small program segments, timers, counters. In this lesson for the first time, we will see that given a practical problem, how to study the problem, what are the steps that you go through to finally arrive at an RLL program.

So, and this will be followed, using a very systematic approach, because as I have already told you that industrial control applications are very critical. In the sense that, if you have programming errors in them, they can be very expensive in terms of money or in terms of even can cost human lives etcetera. So, it is always good to have a very systematic design process by which you can decompose a problem and then finally, arrive at a solution.

(Refer Slide Time: 01:55)



Indian Institute of Technology, Kharagpur

Instructional Objectives

After learning the lesson students should be able to

A. Model simple sequence control applications using State Machines

B. Develop RLL programs for such applications

C. Describe the basic constructs of an SFC

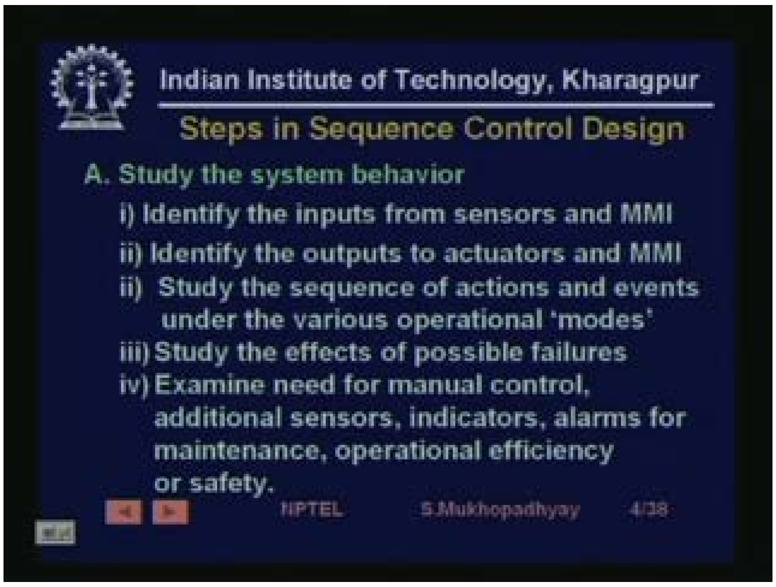
D. Describe two advantages of an SFC

NPTEL S.Mukhopadhyay 3/38

So, we will look at the instructional objectives, the instructional objectives of this lesson are firstly able to model simple sequence control applications using state machines. State machine is a actually formal method and we advocate the use of formal methods, because English can be very ambiguous sometimes contradictory also. So, we have to model it using methods, which are unambiguous consistent do not contain contradictions and are also easy to understand and develop.

Then, from these formal models, we have to develop RLL programs, for such applications. And for doing this, there are certain apart from the RLL programs, there are some modern programming construct, which have been made available. One of them is a SFC or the sequential function chart, so we will take a look at that and also understand some of it is advantages, so these are the instructional objectives of this lesson.

(Refer Slide Time: 02:52)



Indian Institute of Technology, Kharagpur

Steps in Sequence Control Design

A. Study the system behavior

i) Identify the inputs from sensors and MMI

ii) Identify the outputs to actuators and MMI

ii) Study the sequence of actions and events under the various operational 'modes'

iii) Study the effects of possible failures

iv) Examine need for manual control, additional sensors, indicators, alarms for maintenance, operational efficiency or safety.

NPTEL S.Mukhopadhyay 4/38

So, now let us go through the steps in basic broad steps in sequence control design, so first step is to study the system behavior. This is a very critical step and most of the errors that happen in any programming exercise, not only this kind of industrial automation programming, any programming. Mainly arises from the fact, that the programmer or the developer did not understand the system well.

So, this is a very important step and one must, first of all identify inputs to this to the system, that is the controller program, what inputs it will take, inputs can come from either from sensors in the field or it comes from operator interface, which I call the MMI or the Man Machine Interface. So, somebody presses a push button; that is an operator input. On the other hand some limit is which is made, that is a sensor input.

Similarly, identify the outputs, so switch on motor, so motor is an actuator; that is a kind of output. There is another kind of output, for example, switches on some indicator or some lamp that would be an output, which again goes to the MMI or the Man Machine Interface, so we have to first identify these. Then, study the sequence of actions and events, under the various operational modes, this is the main task.

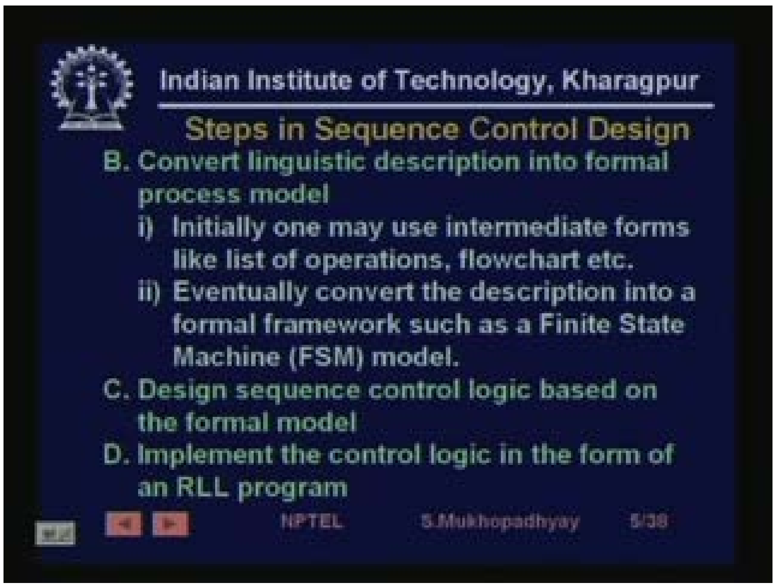
You have to very carefully understand, what is going to happen and what will happen after what, at what time intervals, etcetera. Then, one thing that must be very clearly remembered is that, when you are developing an industrial automation program not only has to remember, not only has to design for normal behavior. But, one must to some extend at least take into account the possible failure, that can occur.

Otherwise, a system that behaves well under normal behavior can behave in a very nasty manner, if some simple element of the system, like a sensor fails. Then, even apart from the automated behavior, one has to examine the requirement; that exists for number one manual control. Manual control is very important, because for finally, if the automation equipment fails, it should be able to operate the system using manual control may be on the field.

While the automated control may be actually working quite a distance away from the actual equipment, it may be housed in some control room. On the other hand, the manual controls may be near the equipment at the field. So, the possibility of including such manual controls must be examined, whether some additional sensors are required, some sensors may be there, but to achieve a kind of functionality, some other sensors may be needed.

Indicators alarms and as well as operational efficiency or safety, these are the factors, which must be considered to finally arrive at the functionality. One must always remember that the customer may not always be able to express his or her needs and a good automation engineer should be able to supplement it, with his own experience in such cases.

(Refer Slide Time: 06:33)


Indian Institute of Technology, Kharagpur

Steps in Sequence Control Design

B. Convert linguistic description into formal process model
   i) Initially one may use intermediate forms like list of operations, flowchart etc.
   ii) Eventually convert the description into a formal framework such as a Finite State Machine (FSM) model.

C. Design sequence control logic based on the formal model

D. Implement the control logic in the form of an RLL program

NPTEL S.Mukhopadhyay 5/38

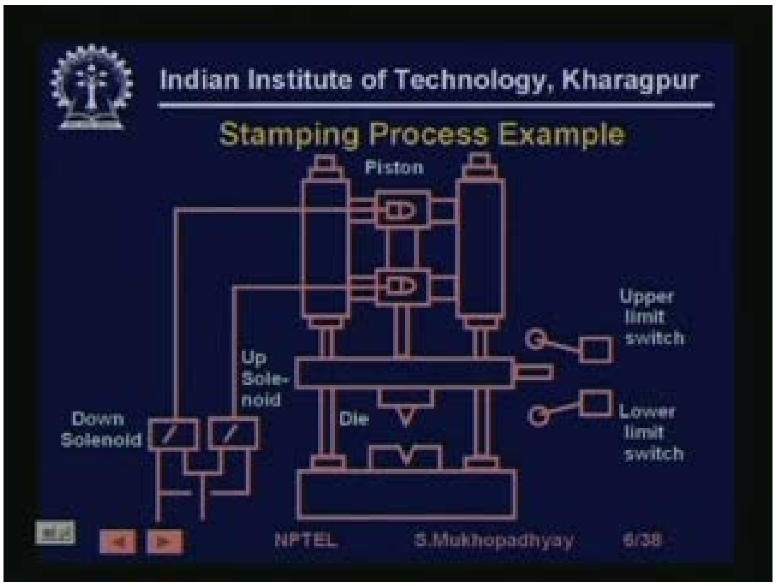
So, having done that, the next step is to convert generally these things are captured manually and using a linguistic description something like you know something like English statement. So, you talk to costumers, talk to engineers on the field and get their requirements, but this is very dangerous to use for program development. So, we have to convert this linguistic description into formal process models.

And in fact, a lot of you know inconsistencies, which are there ambiguities, which are there in the linguistic description, actually surface at this time. Even, for during the process of transforming it into a formal process model 1; may initially use intermediate forms you know like flow charts for example. But, finally it is prescribed that, one should be able to convert it to a formal mathematical frame work, something like, let us say a finite state machine, which we will be using here.

After, up to this the operations are manual; having done that then one has to go for design of the sequence control logic, based on the formal model. And then finally, one has to implement the control logic in the form of an RLL program and it is preferable, that these steps, especially the step D is made as much as much as possible automatic.

Because, this is a step which can be done in an automated manner, once b and c have been carried out and for large programs, it is always preferable to go for automated programming. Because, that will always lead to error free programs, provided your specifications were correct. So, we come back to our old stamping process example, which we have seen in earlier lectures.

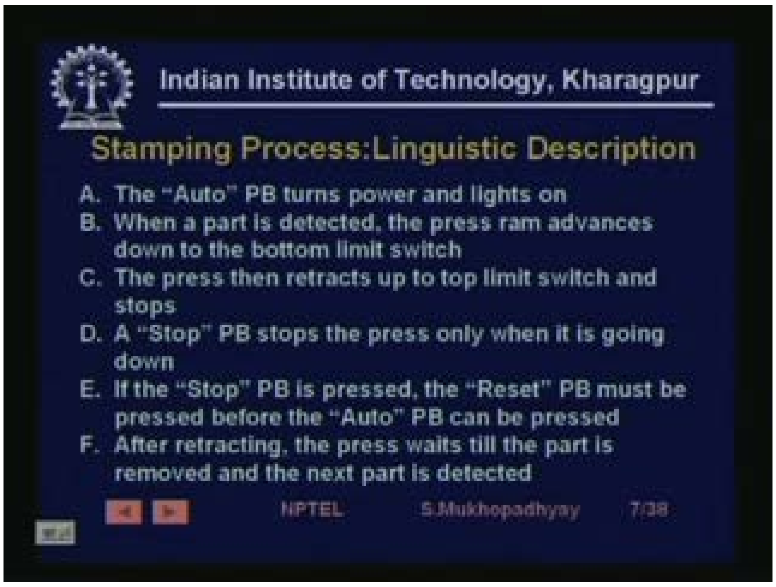
(Refer Slide Time: 08:17)


Indian Institute of Technology, Kharagpur

Stamping Process Example

So, here is a stamping process we know this process, so we have made some addition to it is functionality to be able to explain, you know certain features of a system and make it more complete. So, basic principle is the same, that there is a piston, which and there are two solenoids, hydraulically driven piston, which goes up and down and makes stampings. So, if we know try to write it is list of actions, try to create a linguistic description of the process operation, it looks something like this.

(Refer Slide Time: 08:58)


Indian Institute of Technology, Kharagpur

Stamping Process:Linguistic Description

A. The "Auto" PB turns power and lights on
B. When a part is detected, the press ram advances down to the bottom limit switch
C. The press then retracts up to top limit switch and stops
D. A "Stop" PB stops the press only when it is going down
E. If the "Stop" PB is pressed, the "Reset" PB must be pressed before the "Auto" PB can be pressed
F. After retracting, the press waits till the part is removed and the next part is detected

NPTEL S.Mukhopadhyay 7/38

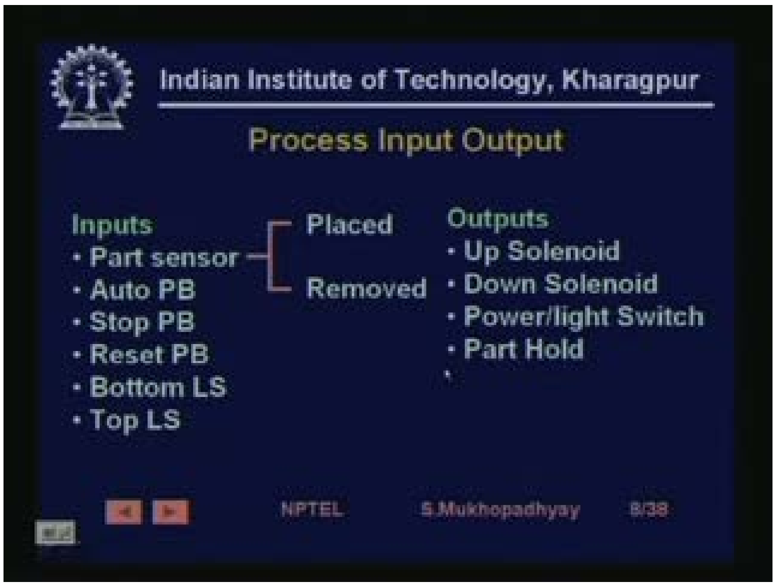
So, in step A, it says that, if auto push button is pressed, so that is an operator input it turns powers and lights on. So, that is possibly a switch once, one or more switches with we will consider it to be one, which will turn the power and the light on one, I mean the moment the auto button is pressed. When a part is detected, the thing to be stamped, when it is detected, it placed at the proper place and detected, so there has to be a part sensor.

The press rams the thing that will be heavy piece, which will move and make a stamping, it advances down and it will stop once, it makes the bottom limit switch. So, that is again a sensor and you must have actuators to make the RAM move down. Then, the press retracts up to the top limit switch and stops, so it makes a stamping and stops. On the other hand, due to some reason, the operator may be like to abort a stamping operation.

So, there is a stop push button provided to the operator and a and a stop push buttons, stops the press, only when it is going down, when it is going up, it has no effect, because anyway, that is not going to cause any problem. If the stop push button has been pressed, it means that something abnormal might have happened. So, the reset push button must be pressed, before the auto push button can be pressed for the next cycle of operation.

So, once you have pressed up, you have to press reset, you know it is a kind of acknowledgment, that the emergency has gone away and the automated operation can resume. Finally, after retracting the after retracting and then going up the press wait is till the part is removed and the next part is detected. So, till the part will be removed and then after that, when the next part will be detected, again the RAM will start coming down, so this is the English behavior of the system. So, now let us try to convert it to an unambiguous mathematical description, so first step as I said is to get the process inputs and outputs.

(Refer Slide Time: 11:22)


Indian Institute of Technology, Kharagpur

Process Input Output

Inputs
- Part sensor — Placed / Removed
- Auto PB
- Stop PB
- Reset PB
- Bottom LS
- Top LS

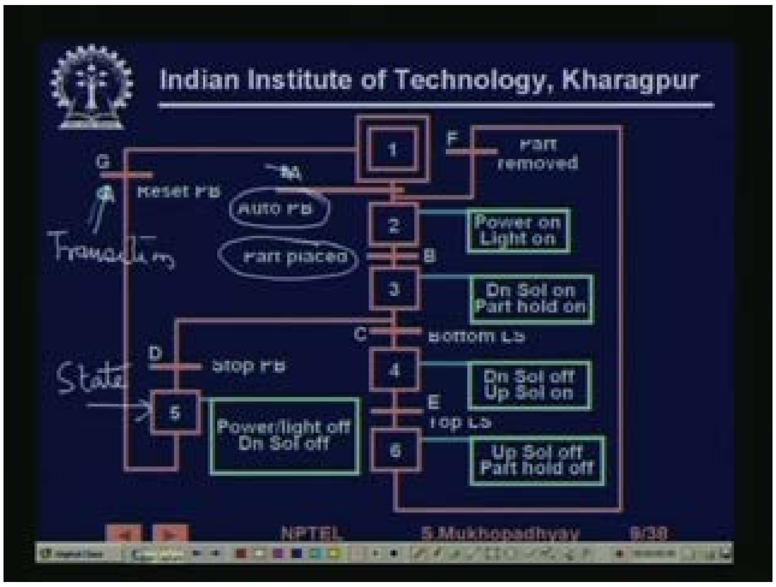
Outputs
- Up Solenoid
- Down Solenoid
- Power/light Switch
- Part Hold

NPTEL S.Mukhopadhyay 8/38

So, what are the process, so here are the process inputs and outputs, so as inputs, we have part sensors, which gives two kinds of it generates two kinds of events, one is part placed, another is part removed. Then, there are three kinds of push button, the auto push button, the stop push button and the reset push button, these three are operator inputs. Then there are the two limits are switch sensors, bottom limit switch and top limit switch.

For outputs, we have four outputs, we have an up solenoid, which moves the RAM up and we have a down solenoid, which moves it down. We have the power light switch and we have a part holder which holds the parts, while it is being stamped. So, these are our outputs. Now, we develop a state machine, so let us try to interpret this diagram.

(Refer Slide Time: 12:20)



So, what is happening in this diagram is that, let me select my pen, so what is happening is that, you see these squares are the states, we are possibly familiar with state machine. So, a state machine is like a graph, which consist of a set of states, these squares are the states and a set of transitions for example, this is a transition, it is not a good color, go back to white. So, this is a transition and this is a state, so this a transition and this is a state.

So, what the system does is that system actually during it is life cycle or during it is activity, the system actually moves from states to states, through transitions. So, it actually spends most of the time in the states and transitions are generally assumed to be momentary, that it is assumed that insignificant amount of time is required to change states. So, you see that it says, that initially, when you have double square, it means that that is a initial state.

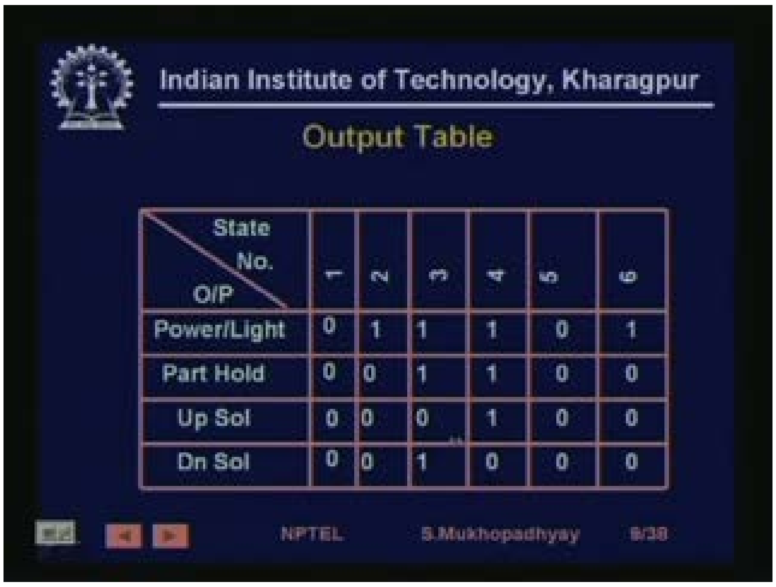
So, if initially, if the auto push button is pressed, this is a transition A, which gets activated, which will take place and take the system from state 1 to state 2, if the auto push button is pressed, so this is the transition condition. You can have much more complicated conditions; in this case we have very simple conditions. And then if this transition occurs, then the system comes to state 2.

In state 2, again if this transition B takes place, whose condition is part, placed it will come to state 3. So, in this way depending on how the sensors are bringing in signals from the field, the various transitions will be enabled and the system will hop from state to state that is the behavior of the system. On the other hand, these green rectangles indicate, that at each state, which are the outputs, which are on for example, you can see that in state 1, nothing is on. None of the outputs are exercised, while in state 2, the power and lights are on. In state 3, the down solenoid is on; actually this is the state, when the solenoid is coming down.

So, you see that this is the initial state here the system is switching on the power and the light and possibly waiting for part placed signal to come. So, it might spend some time here and then it is coming down, so takes times there. Then, from here, it could either go this way or go this way and depending on which one of this has come. So, it may so happen, that the bottom limit switch, if the stop push button has not been pressed.

Then eventually, it will the bottom limit is switch signal will come and then it will come to step 4, in which it will activate these outputs. On the other hand, if before the bottom limit switch is pressed, if the stop push button is pressed, then it will come to this state, where it will simply stop and put the power and light off. So, you see, so this is the way using a graph of nodes and edges, we can describe the behavior of the system unambiguously. So, this is what we know, what we call the state transition diagram and these are the outputs, which are exercised at different.

(Refer Slide Time: 16:15)


Indian Institute of Technology, Kharagpur

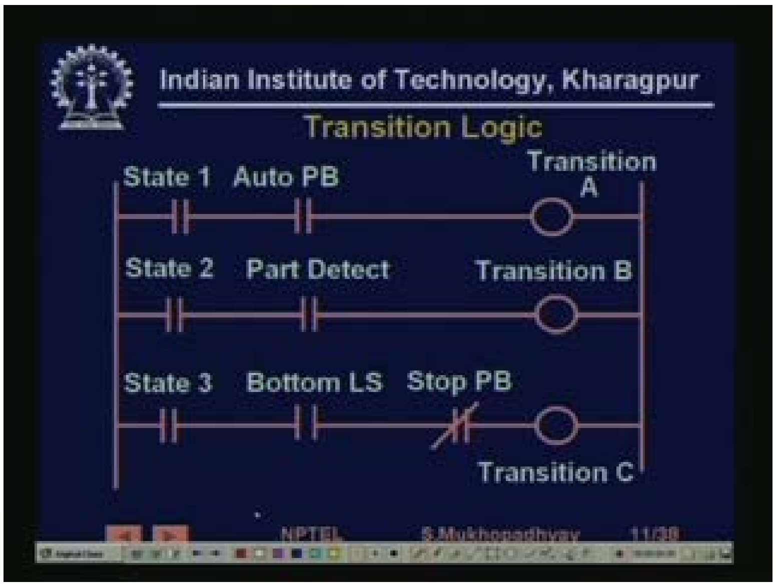
Output Table

State No. / O/P	1	2	3	4	5	6
Power/Light	0	1	1	1	0	1
Part Hold	0	0	1	1	0	0
Up Sol	0	0	0	1	0	0
Dn Sol	0	0	1	0	0	0

NPTEL S.Mukhopadhyay 9/38

And that is actually also in captured, in what is known as an output table, so the output table, says that among the four outputs, that we have namely power, light, switch, part hold, up solenoid and down solenoid. What is their status, whether they are on or off, at in the various states, so there are six states and there are four outputs. So, it says that the power light switch stays on, in state 2, 3, 4 and 6, while the part hold stays on only during 3 and 4, up solenoid is 1, during 4 and down solenoid is 1, during 3, having done that.

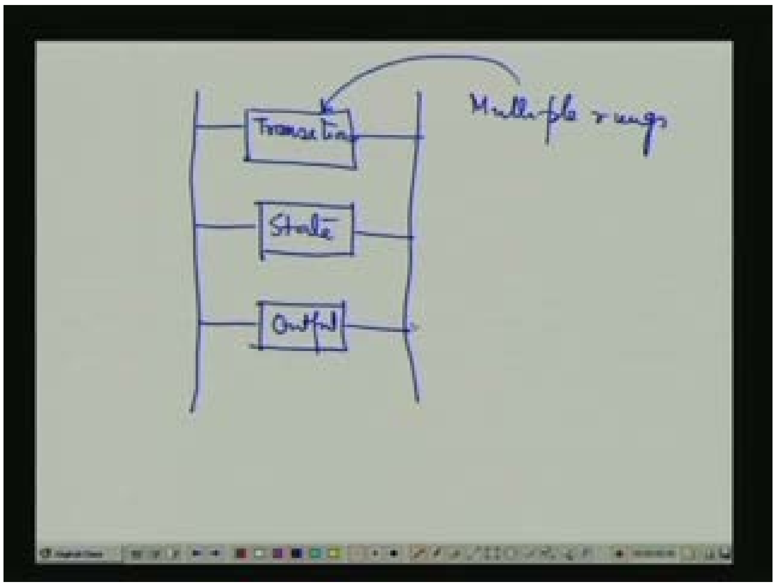
(Refer Slide Time: 16:58)



We can start developing our, we have seen that as the system moves on, the various state logics and transition logics are alternately computed So, first there is a state in which some state logic will be satisfied, depending on that outputs will be exercised, after that at some time transition logic will get satisfied. So, now the system will come to a different state.

So, the previous state logic is going to be falsified and the new state logic will now become true and then based on that the corresponding outputs will get exercised. So, this we have to now capture in a relay ladder logic program. So, we will organize our program into three different blocks, the first block will contain, so maybe I will chose this pen now.

(Refer Slide Time: 18:12)



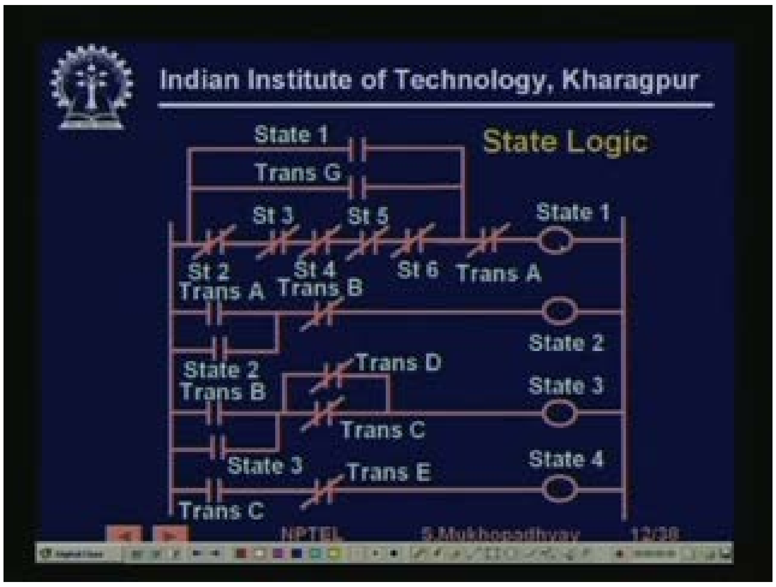
So, the relay ladder logic will consist of three different blocks, the first block will contain the transitions, this is multiple runs, then the state block and finally, the output block. So, we will now describe, these three blocks in the case of this example. So, let us first see the transition logic, for example what does it say, it says that, if when will transition A logic, will be satisfied. The transition A logic, if you recall brings the system from state 1 to state 2.

So, if you wanted to see that, we could go back, just for once, so you see here, transition A takes the system from state 1 to state 2 transition B takes it from state 2 to state 3, transition C and D are in parallel, it could take state 3 to either 4 or 5. So, let us remember this and then go ahead. So, it says that, if the system is in state 1, if it is in state 1; that depends on the state logic.

Then, correspondent to every transition we have an output coil and corresponding to every state we have output coils. So, this is actually an auxiliary contact, corresponding to the output coil called state 1, so it is an abstract variable actually. So, it says that if the system is in state 1, then this contact will be made and at that point of time, if the auto push button signal comes, then transition A will get enabled.

So, it will be on, if we have modeled our system well, then at a time only one transition will get on. If we do not consider concurrency, then at a time only one transition will get on, now what will happen, in the next stage, transition A becomes on and state 1 was already on, so at this point of time, we come to the state logic.

(Refer Slide Time: 20:57)



So, now let us see what happens in the state logic, in the state logic, see state 1 was on, because state 1 was on and because auto push button was pressed, transition A became on, so the computation came from the transition logic to the state logic. So, what happens is that, it founds state 1 is on, so what happened is that, it found that the transition A is on at this point of time it found this transition on.

Because, transition logic has been already evaluated and it has been found to be true, so therefore, this auxiliary contact will be closed. Therefore, now state 2 will be on, now once state 2 is on, two things happen. Firstly, you see in the next, because state 2 is on, this contact will be on and this contact will now be off. So, in the next can cycle, when this runs will be evaluated, this will go off, therefore, because transition A will go off.

So, therefore, state 2, this can go off, it does not matter because this is on, so therefore, this will stay on, it says that, now the system is in state 2. Now, in this way again, when state 2 is on, at that time in the next cycle, some other transition will become enabled, depending on what sensors signals are coming. So, similarly it will turn out, for example in state 3, now you at sometime transition B will take place, so transition B is part placed.

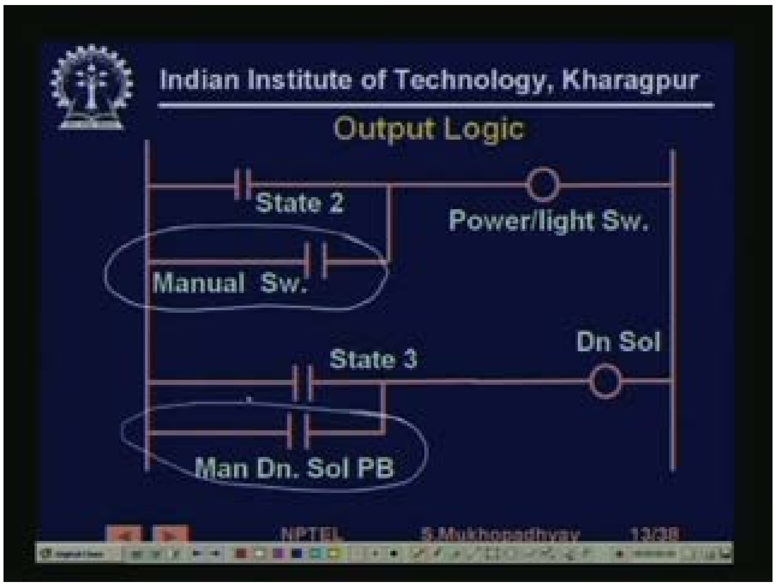
So, when the part will be placed, then if that part placed signal comes, then what will happen is that is the transition B will be on and these are not yet enabled, so therefore, state 3 will be on. On the other hand, while state 3 is on, if either transition C or transition D occurs. Transition D is due to the stop push button being pressed and transition C is due to the bottom limit is which being made.

If any one of these occurs, then it will no longer be in state 3, but it will go to either transit state 4, if transition C occurs state 3 will be falsified and state 4 will become on. On the other hand, if transition D occurs, then this will be falsified and Trans state 5, which is not shown here will become on. So, you see that mechanically, once we have developed the state graph, we can simply mechanically, describe it is behavior.

So, corresponding to every transition, we are going to have one rung, corresponding to every state we are going to have one rung and as I have described, we are going to put the enabling logics. So, we are going to say just from the graph that, if when the system is at state 1, if auto push button is pressed, it will go to state 2. Simple this logic, which is given from the graph will take every transition and will write the corresponding logic in the transition logic.

Similarly, we will say that, if transition x has been enabled, then it will reach this state, so we can do it from the graph, mechanically just one by one, this writing can be actually written by a program itself. So, one need not really think too much about the logic one, one should think about the logic, while he is drawing the diagram, after that the programming becomes automatic, this is very useful. So, now next we will output logic, output logic is very, very simple, especially in this case.

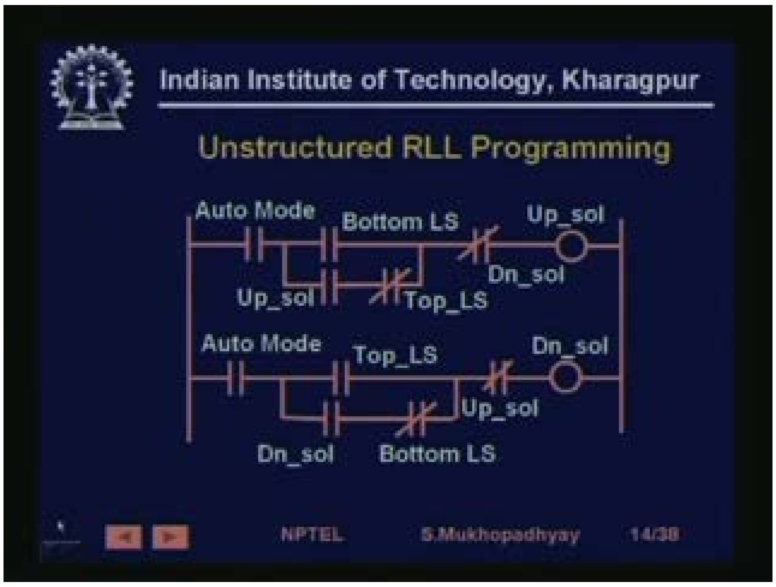
(Refer Slide Time: 25:22)



So, the output logic says, that if you are in state 2, then power light switch should be on as we have given in our output table. So, only thing is that look here, we have also added some manual switch, it can be sometimes, we may need to check, we may need to do things manually also. So, the power light switch will be on here, we have put a manual switch. So, if the PLC is running, that if we press the manual switch, then also power light switch may be made on.

Similarly, we can have a manual down push button, so we can, this is just to demonstrate that, you can put additional logic to include manual operation of the system. Otherwise, this program, simply says that while you are in state 3 down solenoid will have to be activated very simple.

(Refer Slide Time: 26:23)



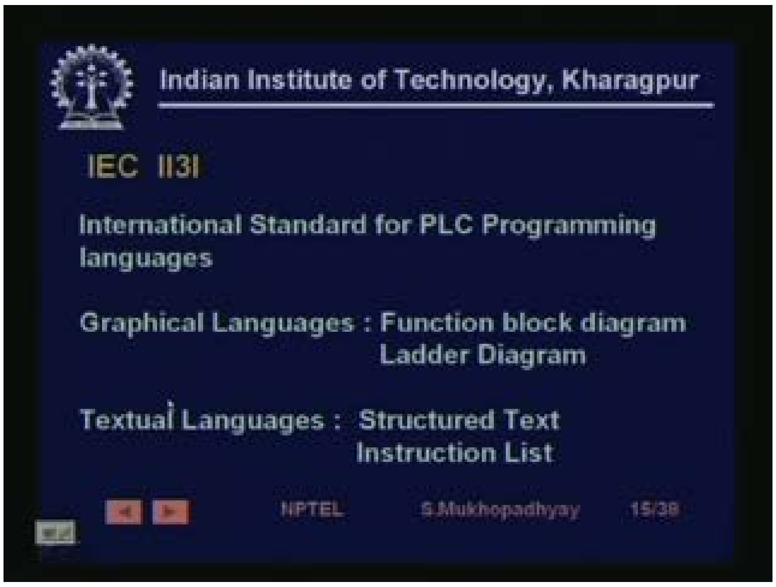
Compare, this with the kind of programs; that we had written earlier, in fact for this process itself, we had written some programs. So, there we did not have any concept of states and transitions, we were directly trying to write outputs in the form of inputs. Now, this kind of problems is that, they are here systems generally have memory, that is why you need the concept state. It is not that, if you get a certain kind of inputs, you will have to produce certain kinds of outputs, it depends on which state the system is in.

So, the concept of state is very important and well you can in bring it possibly in certain cases using some temporary variables, but the kind of here if you look at this program. This program says, it is very complicated logic and I am not even 100 percent sure, it is very difficult to be 100 percent sure, whether this logic is pull proof. It says that, if the auto mode by the way, this auto mode is actually you can, it is an auxiliary contact.

Corresponding to some logical variable, which you can set by a simple rung, that if it is auto P B and then you have an auto mode coil and then you have this is auto P B and here you can have auto mode. So, you can have a auto mode coil and this will be an auto mode auxiliary switch, so that the PB can be released, so this is a sort of a you know persistent input.

So, if this auto mode is on, so it says that, everything will work only, if the auto mode is on and then if the bottom limit switch is made and the down solenoid is not on, then the up solenoid can be energized. Similarly, and once the up solenoid is energized, it will remain energized until the top limit switch is looks. But, one is never sure and especially when problems will have 200 states, then it will be impossible to write such direct programs. Where the chances will be very high, that if somebody, wants to write it, he will make mistakes. So, this is a typical example of unstructured programming, the kind of things that we did before this lesson.

(Refer Slide Time: 28:50)


Indian Institute of Technology, Kharagpur

IEC 1131

International Standard for PLC Programming languages

Graphical Languages : Function block diagram
Ladder Diagram

Textual Languages : Structured Text
Instruction List

NPTEL S.Mukhopadhyay 15/38

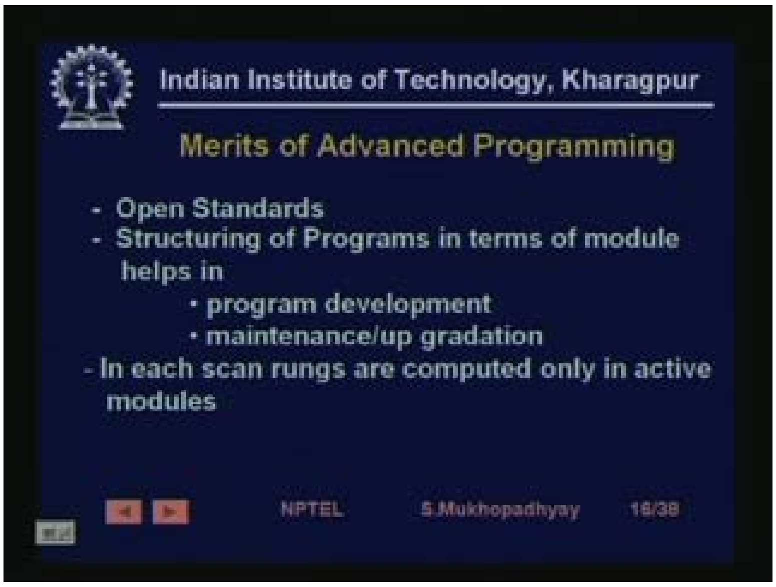
Now, having said these things, so this is in very brief, this is a structured design approach to other programming. Now, we must mention, since we are coming to the end of this programming part, we must mention that there are you know certain standards of programming languages, which have come for example, IEC 1131, IEC stands for the International Electro Technical Commission and the 1131 is a number.

So, this is an international standard for PLC programming and it classifies languages into two types, one are graphical languages, 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, 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. So, for now we have understood that, this state based design is very useful. So, therefore to be able to capture this state transition logic, a separate language has been proposed.

Separate methods of advanced programming have been proposed and we are going to take a brief look at that. So, some of the merit is of this kind of you know, advanced programming which takes this kind of abstract state transition logic. Firstly, because they had open standards in the sense, that anybody can write a program, which can be used by somebody else.

(Refer Slide Time: 30:42)


Indian Institute of Technology, Kharagpur

Merits of Advanced Programming

- Open Standards
- Structuring of Programs in terms of module helps in
    - program development
    - maintenance/up gradation
- In each scan rungs are computed only in active modules

NPTEL S.Mukhopadhyay 16/38

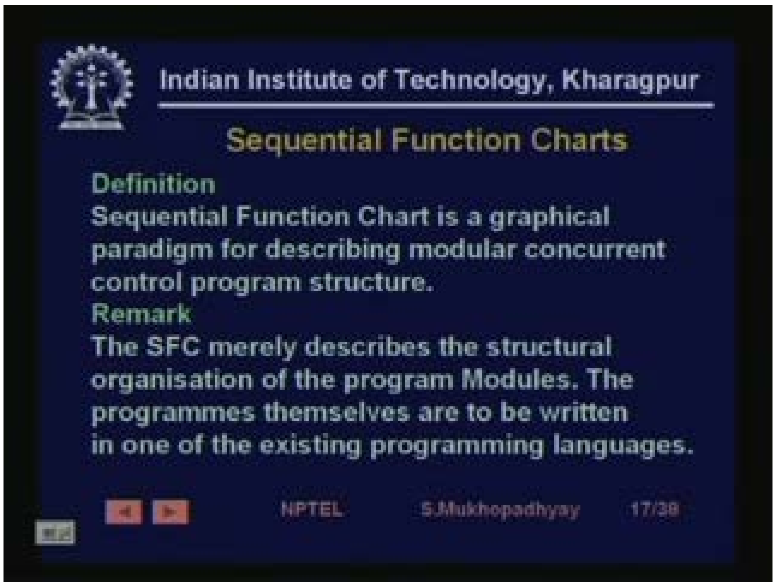
Then structuring of programs in terms of module helps in program development and program maintenance, very important program up gradation, you want to add a new functionality, you will find that it is very simple, just change the state diagram. And then find out that, now in the new state diagram, if you have three extra states, simply you have to add those corresponding rungs.

If you have some extra transitions, you have to add those rung and if you have to redirect some transitions to now new states, then you have to code that logic, that transitions 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 benefit is of structured programming.

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 rungs 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 and it supports concurrency, that is very important. That is about concurrency, because it very often happens, there are several things, which will be taking place together.

And which are best modeled as concurrent and especially now, that we have you know these. So, called multitasking operating systems, all are executives running on microprocessors, so there is no reason, why this cannot be done, I mean we have the technology to enable it. So, therefore we must use it, we have formalism, which is the standard and which are called sequential function charts.

(Refer Slide Time: 32:38)


Indian Institute of Technology, Kharagpur

Sequential Function Charts

Definition

Sequential Function Chart is a graphical paradigm for describing modular concurrent control program structure.

Remark

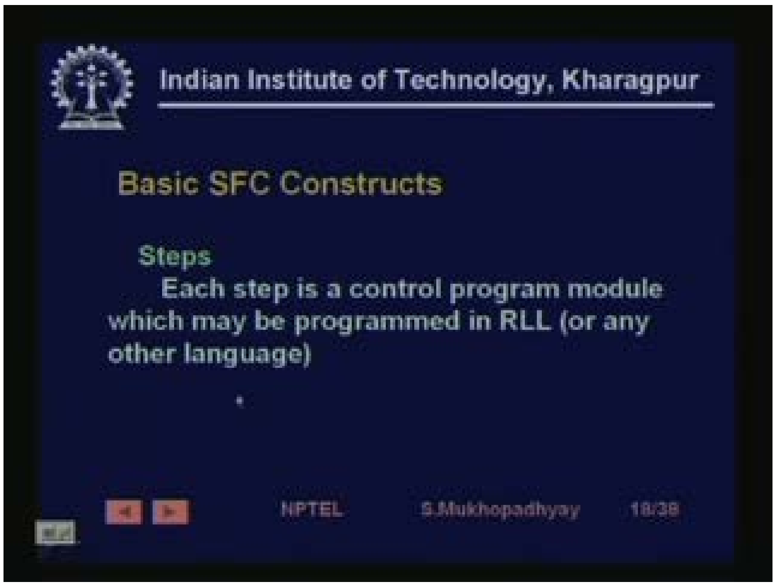
The SFC merely describes the structural organisation of the program Modules. The programmes themselves are to be written in one of the existing programming languages.

NPTEL S.Mukhopadhyay 17/38

So, a sequential function is actually a graphical paradigm, for describing modular concurrent control program structure, so remember that, it is basically used to describe structure and not the program itself. So, 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 this functions will be executed when and under what condition.

So, you have this remark which says that the SFC; merely describes the structural organization of the program modules. While, 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: 33:35)



Indian Institute of Technology, Kharagpur

Basic SFC Constructs

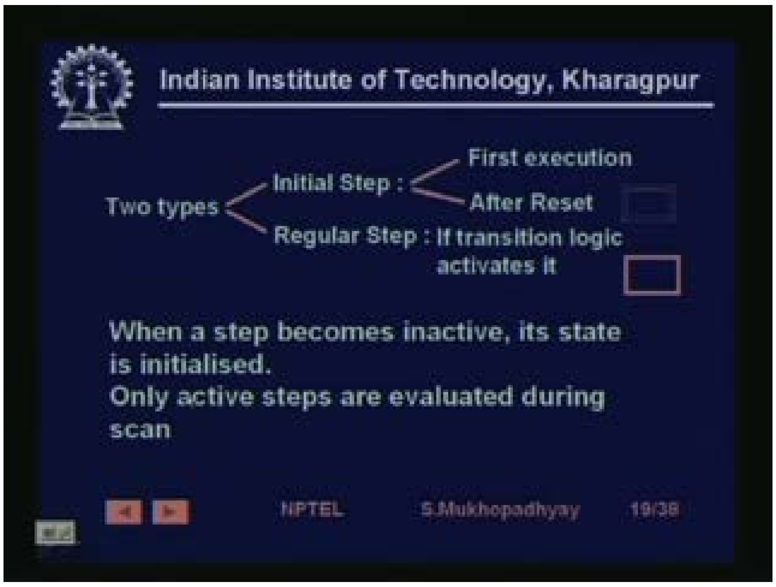
Steps

Each step is a control program module which may be programmed in RLL (or any other language)

NPTEL S.Mukhopadhyay 18/38

So, if let quick look at the basic SFC constructs, they are basically state machine like constructs and contains some constructs, which are simple programming constructs with which we are familiar. 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 RLL or any other language, so as I said it is a function.

(Refer Slide Time: 34:02)


Indian Institute of Technology, Kharagpur

Two types:
- Initial Step:
  - First execution
  - After Reset
- Regular Step: If transition logic activates it

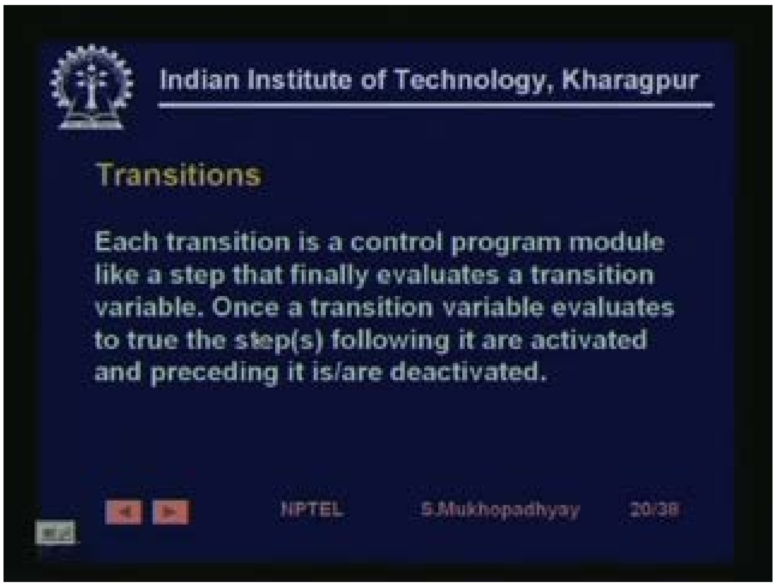
When a step becomes inactive, its state is initialised.
Only active steps are evaluated during scan

NPTEL S.Mukhopadhyay 19/38

Similarly, this steps can be of two types, number 1, initial step I have marked the initial step and the initial step execution can be of two types. One, when the first time, they 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 the programs.

So, after reset also you can execute a particular type of step called the initial step, typically used for initializing variables and steps and intermediate variables. Otherwise, you have regular steps and which one of them are active at any time, actually depends on the transition logic as we have seen. So, when a step becomes inactive, it is state is initialized and only active steps are evaluated during the scan that saves time.

(Refer Slide Time: 34:59)


Indian Institute of Technology, Kharagpur

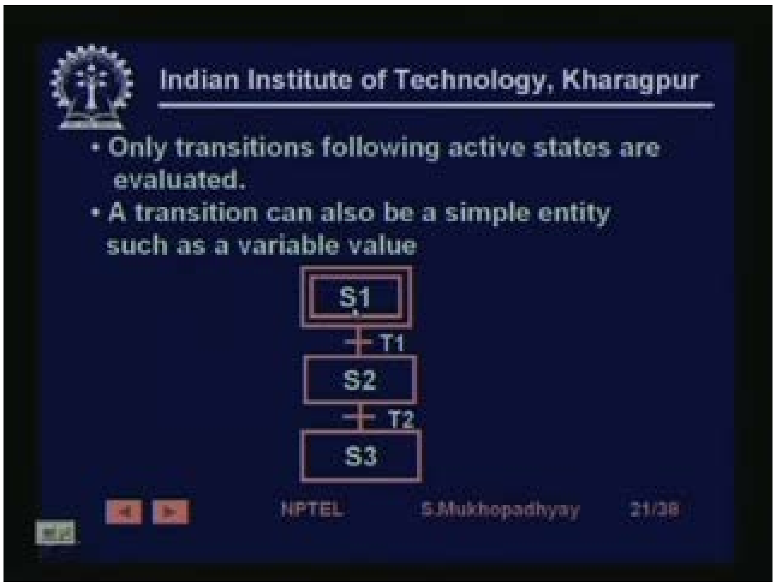
Transitions

Each transition is a control program module like a step that finally evaluates a transition variable. Once a transition variable evaluates to true the step(s) following it are activated and preceding it is/are deactivated.

NPTEL S.Mukhopadhyay 20/38

Now, 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 a transition variable evaluates to true, then the steps following it are activated and we say steps, because we in this formalism. Typically, we have seen thaT 1 transition can go to only one step, provided you do not have concurrency, but since SFC’s support concurrency. So, it may happen, that after a one transition, two concurrent you know threads of execution will start. So, that is why we have said steps following it are activated and those preceding it are deactivated.

(Refer Slide Time: 35:46)


Indian Institute of Technology, Kharagpur

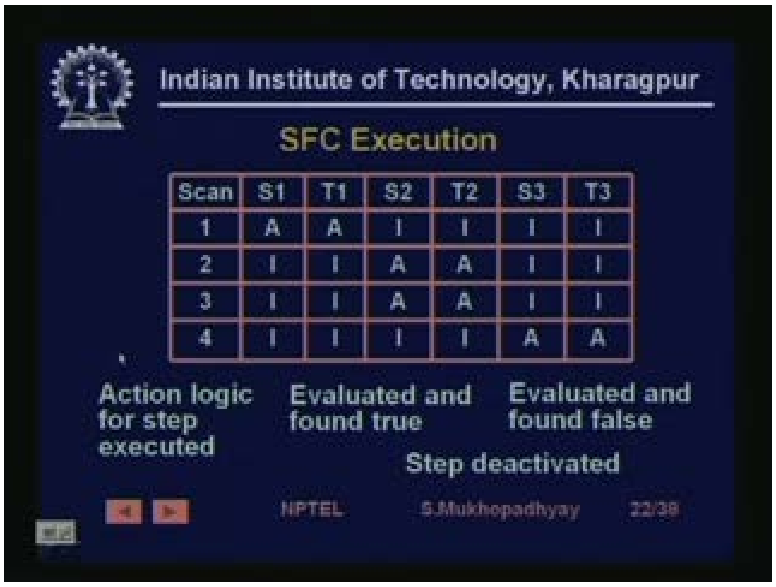
- Only transitions following active states are evaluated.
- A transition can also be a simple entity such as a variable value

NPTEL S.Mukhopadhyay 21/38

Only transition following active states, so again when you in a state already, which is active, only certain transitions can occur, so therefore, only those transition need to evaluated, so we only those are evaluated. A transition 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.

So, for example, in this case, it says that S 1 is actually an initial step and then while S 1 is active, T 1 is active and if the conditions for T 1 are fulfilled. They will come to S 2, if T 2 will become also active and if T 2 occurs, then it will go to S 3.

(Refer Slide Time: 36:41)



Indian Institute of Technology, Kharagpur

SFC Execution

Scan	S1	T1	S2	T2	S3	T3
1	A	A	I	I	I	I
2	I	I	A	A	I	I
3	I	I	A	A	I	I
4	I	I	I	I	A	A

Action logic for step executed

Evaluated and found true

Evaluated and found false

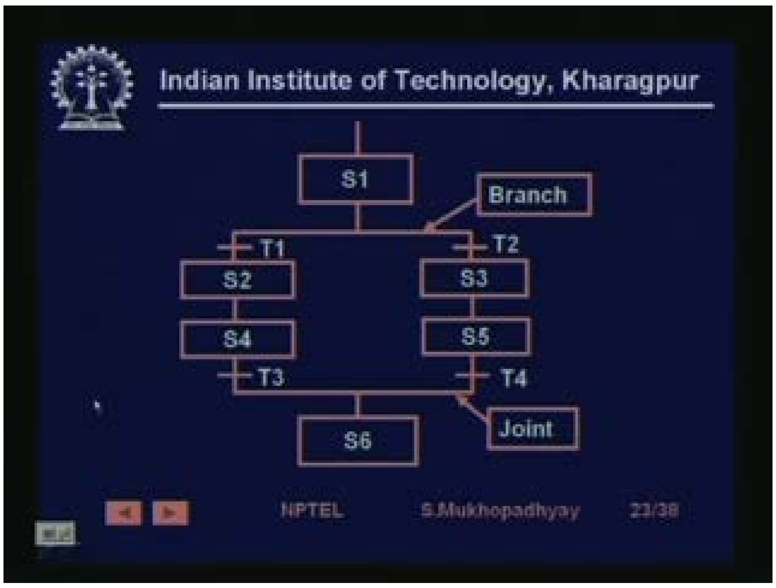
Step deactivated

NPTEL S.Mukhopadhyay 22/38

So, you see that, if you see the activations in scan 1, it may so happen that S 1 and T 1 are active, while in scan 2, it may happen that T 1 has fired, once T 1 has fired, that is actually taken place, because T 1 is active. So, it has come to S 2 and then T 2 is active, so if T 2 active, it is continuously evaluated, it may happen that in scan 3, T 2 is evaluated to be false.

So, these two continue to be active, while at scan 4, it may happen that T 2 is now actually become true, so therefore S 3 has become active and then some A transition outgoing from it will become active. So, in this way certain parts of the program at a time, becomes active as the system moves through the sequential function chart.

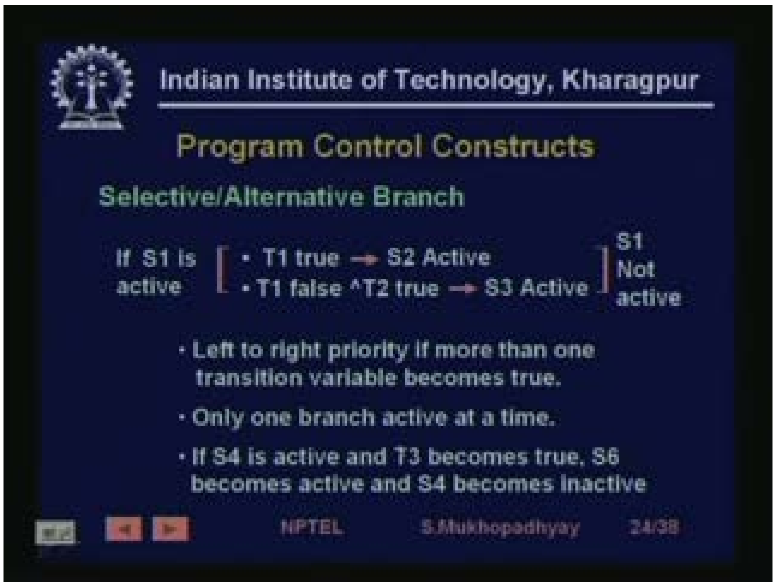
(Refer Slide Time: 37:36)


Indian Institute of Technology, Kharagpur

NPTEL S.Mukhopadhyay 23/38

For example there are you can describe several kinds of you know programming constructs for example, this is a case, where the system if at S 1, at while the system is at S 1, T 1 and T 2 are active. So, any one of them can become true and if T 1 is true, S 2 the next active state will be S 2. If T 2 is true, next active state will be S 3, while it could conceivably also happen that T 1 and T 2 in a given stand T 1 and T 2 both become active. So, in that case you have to dissolve, because both transitions cannot take place simultaneously. So, there is a convention that the left most transition becomes actually takes place. So, if T 1 and T 2 will both become active true at a same time then T 1 is assumed to have taken place.

(Refer Slide Time: 38:44)


Indian Institute of Technology, Kharagpur

Program Control Constructs

Selective/Alternative Branch

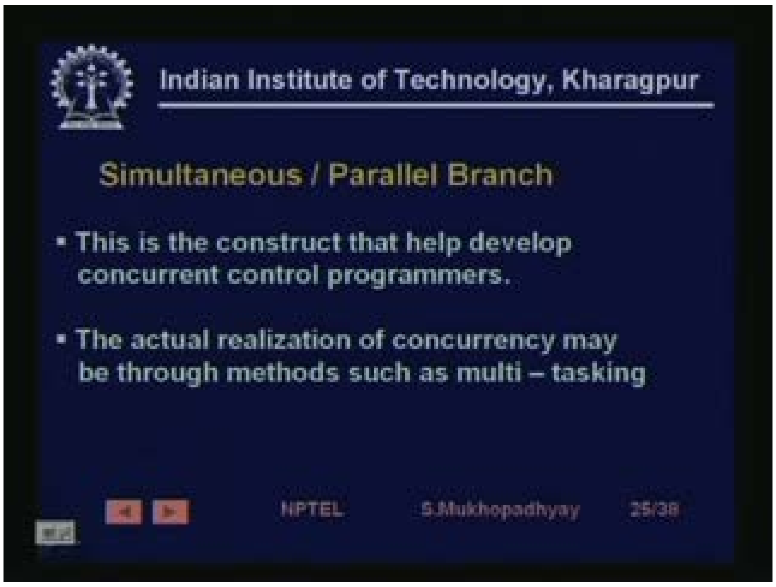
If S1 is active [• T1 true → S2 Active
• T1 false ^T2 true → S3 Active] S1 Not active

- Left to right priority if more than one transition variable becomes true.
- Only one branch active at a time.
- If S4 is active and T3 becomes true, S6 becomes active and S4 becomes inactive

NPTEL S.Mukhopadhyay 24/38

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. So, as I said, that if S 1 is active, T 1 is true and then S 2 becomes active, if T 1 is false and T 2 is true, then S 3 becomes active and then the moment S 3 becomes active, S 1 become inactive. Similarly, left to priority, as I said and only one branch can be active at a time, if S 4 is active and T 3 becomes true, S 6 becomes active and S 4 becomes inactive same thing.

(Refer Slide Time: 39:23)



Indian Institute of Technology, Kharagpur

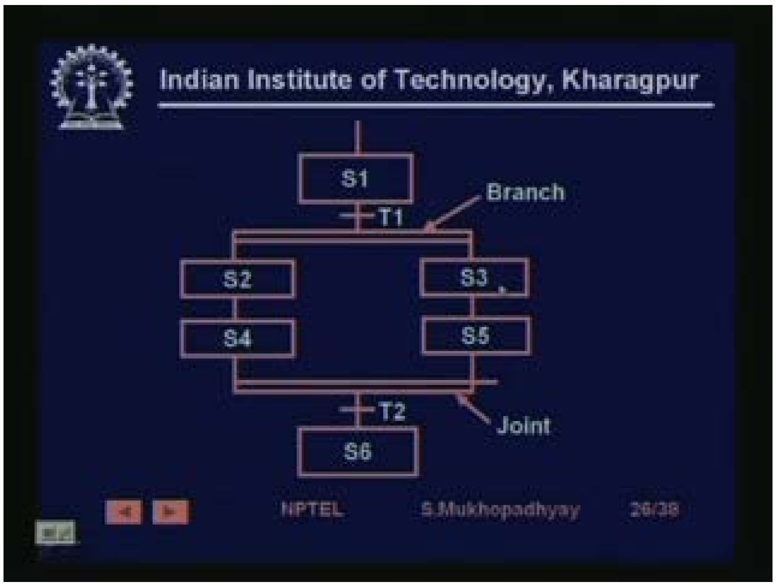
Simultaneous / Parallel Branch

- This is the construct that help develop concurrent control programmers.
- The actual realization of concurrency may be through methods such as multi – tasking

NPTEL S.Mukhopadhyay 25/38

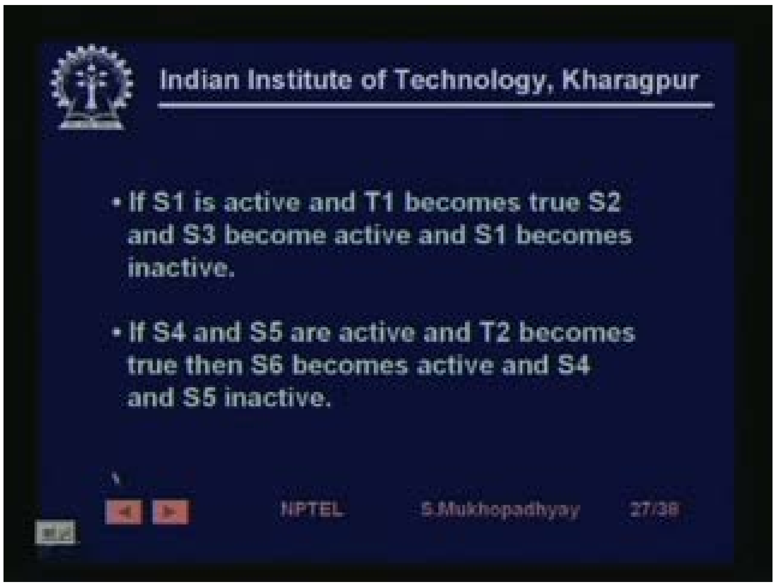
So, the similarly just like selective or alternative branches you can also have simultaneous or parallel branches So, this is a new construct, which is not possible to be you implement in a normal state machine, so this is the construct that helps concurrent control programs. The actual realization, how you will rung concurrent executions, it can be made using various you know operating system, methodology such as multitasking.

(Refer Slide Time: 39:55)



So, here you have a parallel branch, when you see that, at S 1, if T 1 occurs, then both these states become active together and then at some point of time S 4 becomes active, but T 2 cannot take place, unless this branch also comes to S 5. So, only when S 4 and S 5 will become active at that time T 2 will become active and it will be evaluated. If it evaluation is true, then it will come to S 6 and then S 4 and S 5 will become inactive.

(Refer Slide Time: 40:30)



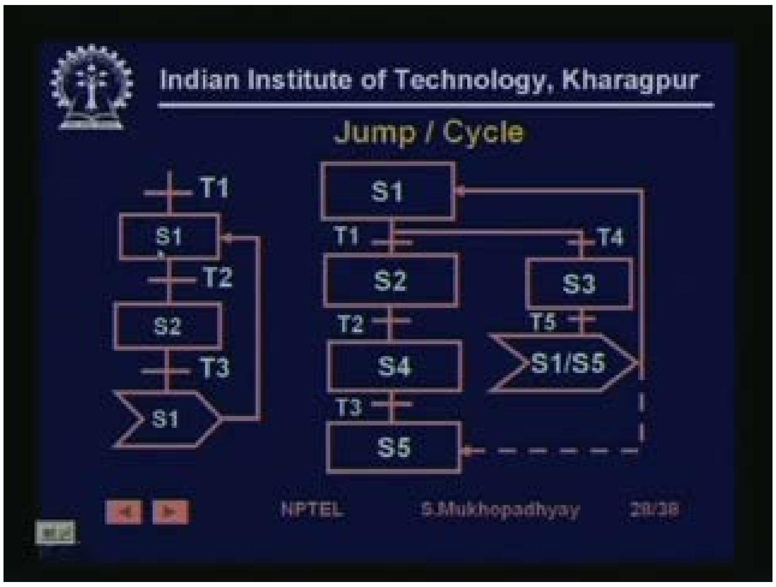
Indian Institute of Technology, Kharagpur

- 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.

NPTEL S.Mukhopadhyay 27/38

So, that is same thing is written here, if S 1 is active and T 1 becomes true, S 2 and S 3 become active and S 1 becomes inactive. If S 4 and S 5 are active and T 2 becomes true, then S 6 becomes active and S 4 and S 5 inactive.

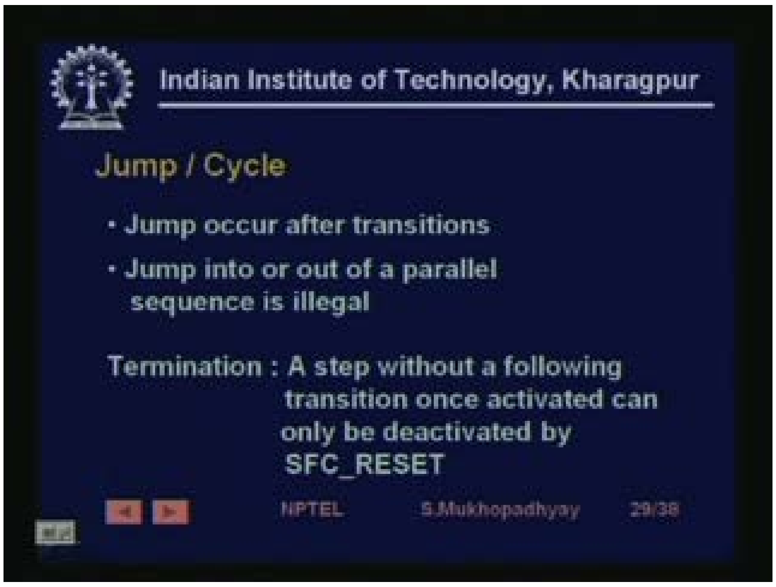
(Refer Slide Time: 40:48)


Indian Institute of Technology, Kharagpur

Jump / Cycle

Similarly, you can have among program control statements, for example, you have a jump, where after S 1, S 2 after T 3, it will go back to S 1. Similarly, you can have a jump inside the loop, you can free, I mean jumping here you can either come to S 5 or you can come to S 1. So, basically you can execute these kinds of program control statements using SFCs.

(Refer Slide Time: 41:15)


Indian Institute of Technology, Kharagpur

Jump / Cycle

- Jump occur after transitions
- Jump into or out of a parallel sequence is illegal

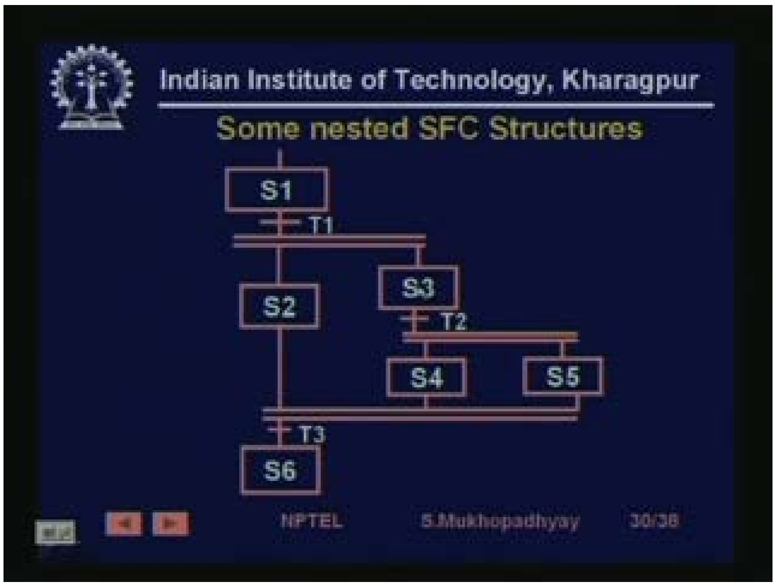
Termination : A step without a following transition once activated can only be deactivated by SFC_RESET

NPTEL S.Mukhopadhyay 29/38

So, a jump cycle jump occurs after transitions and jump into or out of a 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 may be in different states at different arms, but you must enter them together and you must leave them together.

Similarly, last you may have termination of a program, so step without a following transition, if you have a step out of which there is no transition; that is a dead state. And it once, it is activated; it can only be deactivated by a special instruction called SFC reset.

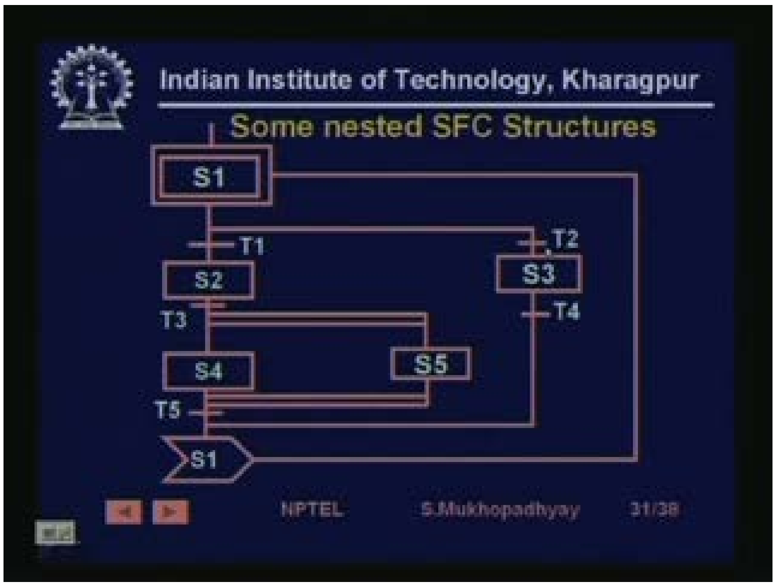
(Refer Slide Time: 41:53)


Indian Institute of Technology, Kharagpur

Some nested SFC Structures

You can nest programming, you can nest within a concurrent loop, so here if T 1 occurs, S 2 and S 3, then from S 3, if T 2 occurs, then simultaneously S 2, S 4 and S 5 can become active and if the all of them are active, then T 3 will take place and then S 6.

(Refer Slide Time: 42:18)

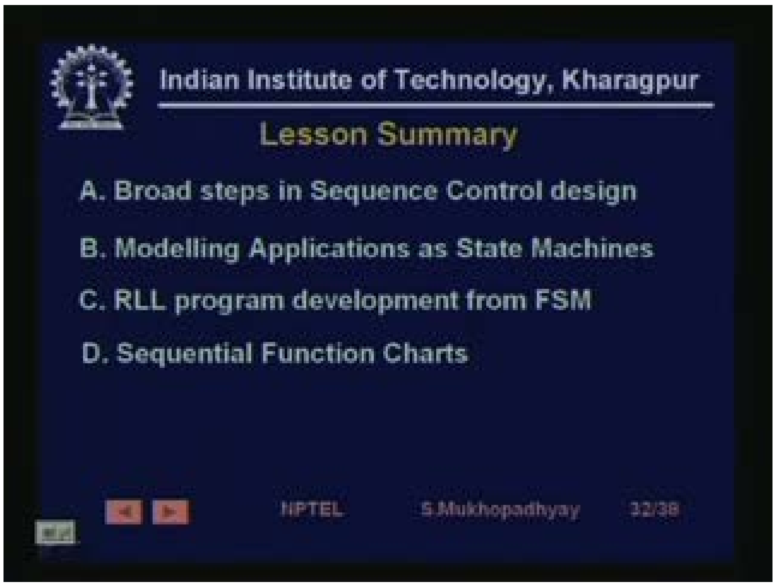

Indian Institute of Technology, Kharagpur

Some nested SFC Structures

NPTEL S.Mukhopadhyay 31/38

Similarly, you see here what is happening, exactly the same thing is happening; here you have a selective branch, so you can either go into S 2 to T 1 or go into S 3 through T 2, that is a selective branch. While you are in a selective branch, you can execute a parallel branch and then if when S 4 and S 5 are both active at that time. If T 2 evaluates to true, then you will come to a jump, which will again take you to S 1.

(Refer Slide Time: 42:59)


Indian Institute of Technology, Kharagpur

Lesson Summary

A. Broad steps in Sequence Control design

B. Modelling Applications as State Machines

C. RLL program development from FSM

D. Sequential Function Charts

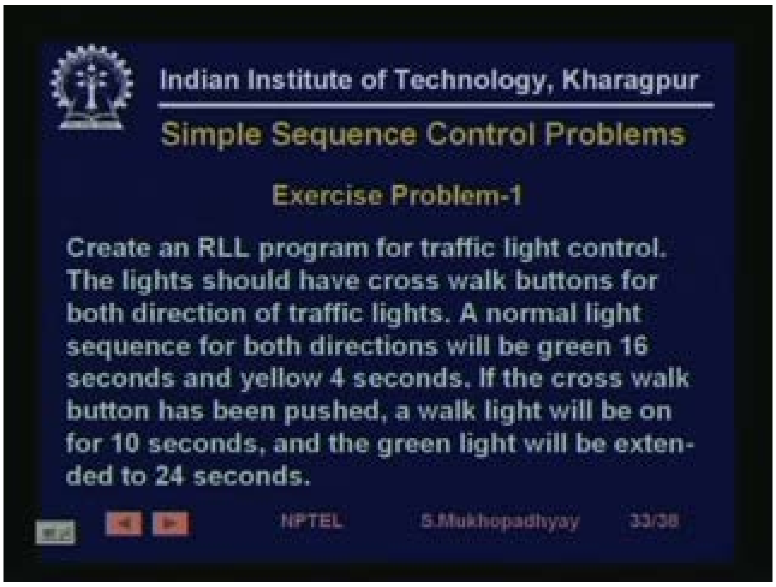
NPTEL S.Mukhopadhyay 32/38

So, such control flows, you can specify, so we have to come to the end of the lesson, in this lesson, what have we learnt, we have learnt the broad steps in the sequence control design and these are very important that first of all identify the inputs and outputs. Then very critically study the system behavior, look at the requirements for manual control operator safety, faults etcetera and then try to formalize this description.

So, first may be write the operation in crisps steps in English and then try to convert this description of steps into some short of formalism. And we have in this lecture, we have seen the formalism of sequential, I mean state machines which can be programmed using a sequential function chart. 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 is state machine.

And then we have shown that, how given the state machine, how very mechanically, you can arrive at 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 a very much expected to lead to you know error free programs. And finally, we actually found syntax, I mean some language graphical language, which can actually capture this flow of logic in terms of states and transitions. So, this program control or the program flow description strategy, using sequential function charts, we have seen, now before ending, I think it is nice to look at some problems.

(Refer Slide Time: 45:19)


Indian Institute of Technology, Kharagpur

Simple Sequence Control Problems

Exercise Problem-1

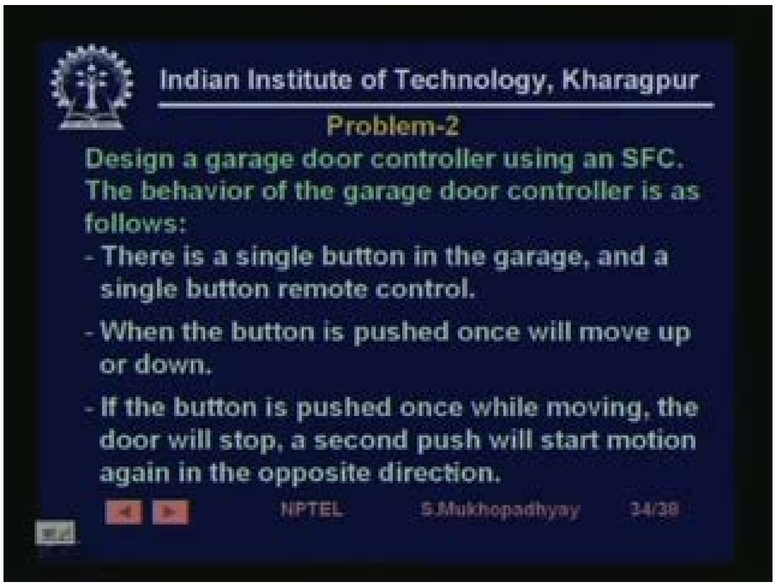
Create an RLL program for traffic light control. The lights should have cross walk buttons for both direction of traffic lights. A normal light sequence for both directions will be green 16 seconds and yellow 4 seconds. If the cross walk button has been pushed, a walk light will be on for 10 seconds, and the green light will be extended to 24 seconds.

NPTEL S.Mukhopadhyay 33/38

For your examples, you can try to create, let us looks at exercise problem number 1, so the problem number one says create an RLL program, for traffic light control. So, what happens, that the lights should have cross walk buttons, for both directions of traffic light. A normal light sequence for both directions will be green, 16 seconds and yellow, 4 seconds.

If the cross walk button has been pushed, then a walk light will be on for 10 seconds and the green light will be extended to 24 seconds, so see this much of description is only given now while, what is the task. The first task would be to develop a state machine description and probably write it in terms of a SFC and then while going to do this SFC we will actually encounter several problems. For example, we may find that things are not set, for example let us look at the SFC of this problem.

(Refer Slide Time: 46:51)



Indian Institute of Technology, Kharagpur

Problem-2

Design a garage door controller using an SFC. The behavior of the garage door controller is as follows:

- There is a single button in the garage, and a single button remote control.
- When the button is pushed once will move up or down.
- If the button is pushed once while moving, the door will stop, a second push will start motion again in the opposite direction.

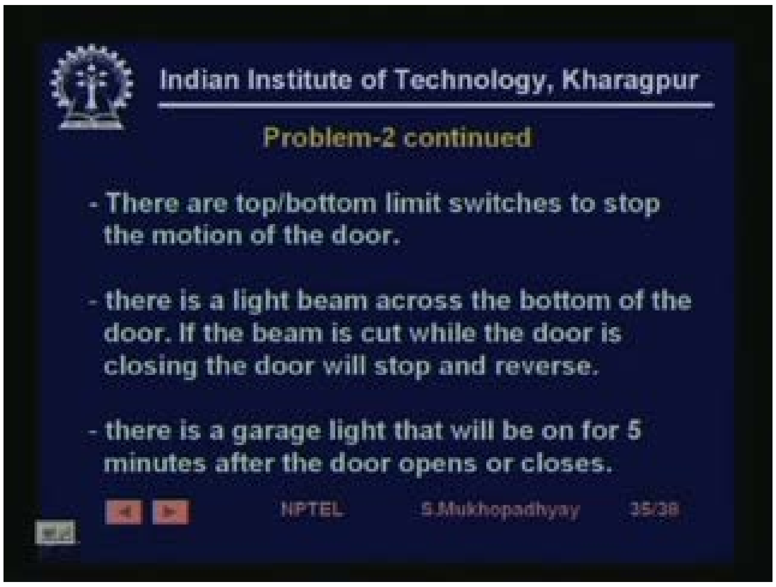
NPTEL S.Mukhopadhyay 34/38

So, we will for the time being we will skip problem number 2 or let us look at problem number 2; next and then we will go to the SFCs. 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 and then the door 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 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 same button, but sometimes it is stopping, sometimes it is moving up, sometimes it is 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 at state, as you might have noticed, when you have design digital logic circuit. So, you have sequential, whenever you have memory, you have sequential logic.

(Refer Slide Time: 48:30)



Indian Institute of Technology, Kharagpur

Problem-2 continued

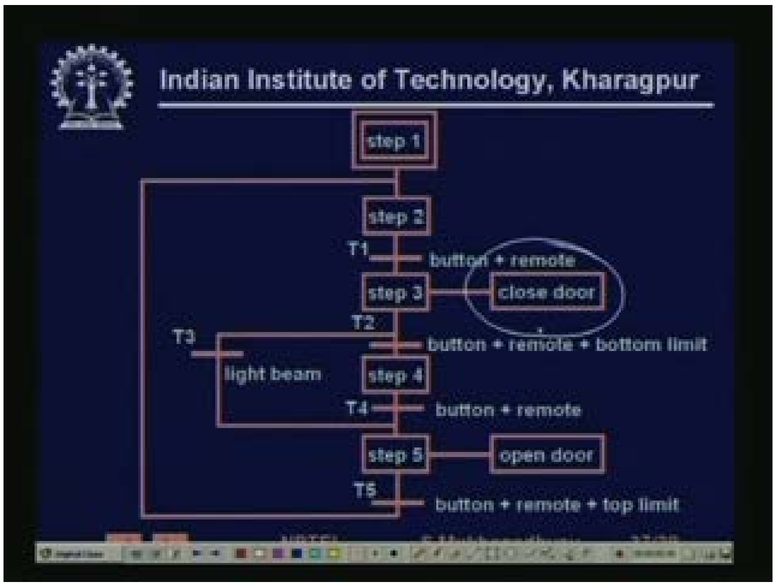
- There are top/bottom limit switches to stop the motion of the door.
- there is a light beam across the bottom of the door. If the beam is cut while the door is closing the door will stop and reverse.
- there is a garage light that will be on for 5 minutes after the door opens or closes.

NPTEL S.Mukhopadhyay 35/38

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 is switches, there is a light beam across the bottom of the door, if the beam is cut, while the door is closing, then the door will stop and reverse. So, while the door is closing, suppose if you put something below it that is trying to ensure that the door can actually close.

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 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 there is clearance. Otherwise, that if the light will be interrupted at any time, immediately the door will stop and reverse, it will think that there is something it cannot close it. There is a garage light that will be on for 5 minutes, after the door opens or closes, so you have to use a timer here. So, any time you do an operation, the light is going to go on and stay 5 minutes.

(Refer Slide Time: 50:01)



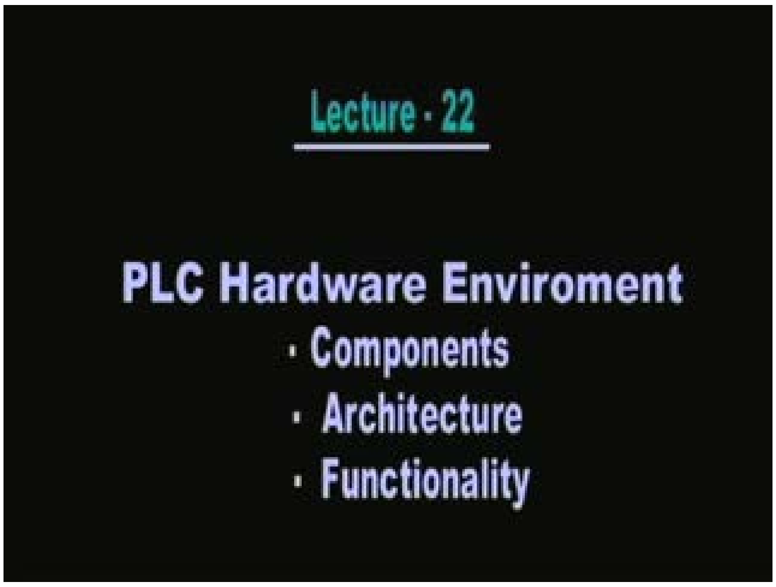
So, let us first look at the next which is simple, so here you have the garage door, so let us see how they modeled it, so you have in this case, if button is pressed and so you have step 1 going to step 2 going to step 3, this is the close door. So, let us go back, so there is single button in the garage and a single button remote control. So, you can have either a button pressed in the garage or you can have a button pressed from the remote.

So, there are two kinds of buttons, so you have either a button pressed from the garage or from the remote, then we will go to step three and it starts closing the door, this the output. On the other hand, if button has been, if from there, if either a button has been pressed, either from local or from remote or if we press a button, what will happen, then the door will stop.

On the other hand, if the limit is switch is made, what will happen the door will stop, 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, 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 we have captured the behavior of the garage door in this form using SFC, you can do a similar thing also for the traffic light, let that bearings and exercise, that brings us to the end of this lesson, thank you very much and see you again for the next lesson.

(Refer Slide Time: 52:34)



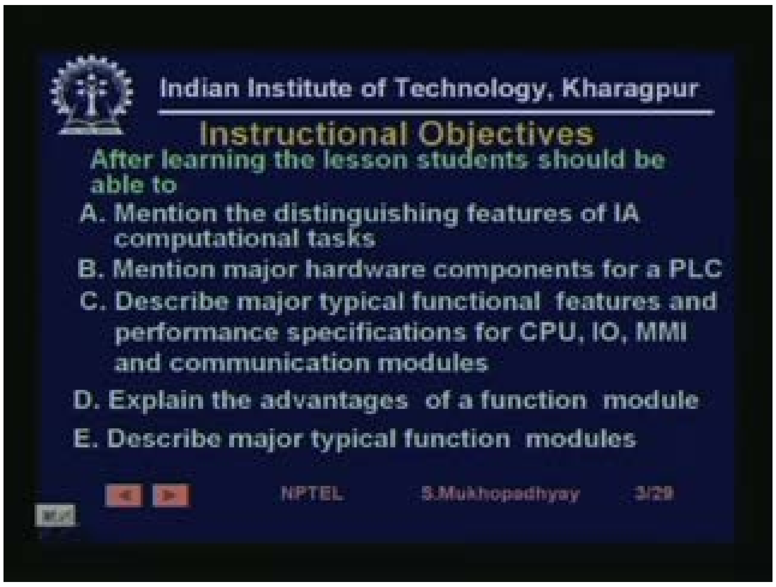
Lecture - 22

PLC Hardware Enviroment

- Components
- Architecture
- Functionality

Welcome to lesson 22, so far we have learnt about the basic functioning of a PLC, we have learnt, how to write a program for it. But, all these time, we have seen the PLC as an abstract device, we have not seen, what is inside a PLC system, what actually physically makes it. So, in this lesson, we are going to look at the hardware environment of the PLCs. Basically, what PLC systems are made of, so we are going to look at components of the PLC connected and their functionality, that is describing, what they do, let us see before we begin, it is customary to see the instructional objectives.

(Refer Slide Time: 54:07)



Indian Institute of Technology, Kharagpur

Instructional Objectives

After learning the lesson students should be able to

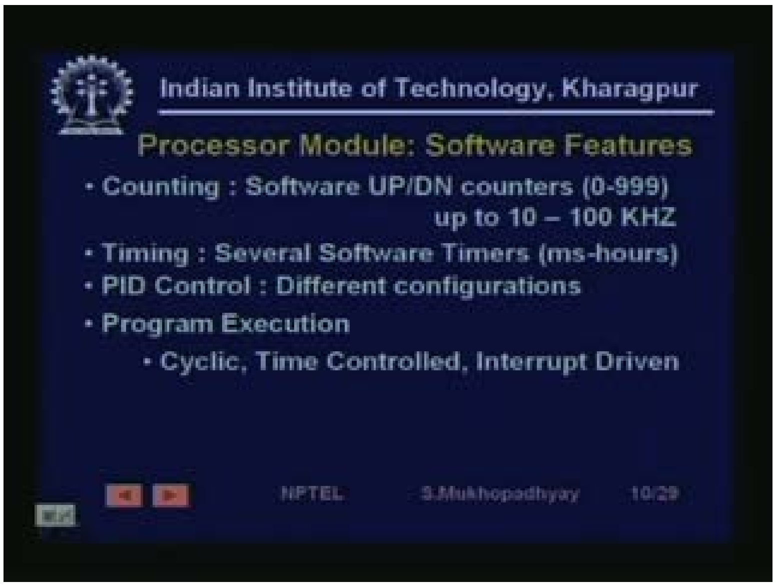
A. Mention the distinguishing features of IA computational tasks
B. Mention major hardware components for a PLC
C. Describe major typical functional features and performance specifications for CPU, IO, MMI and communication modules
D. Explain the advantages of a function module
E. Describe major typical function modules

NPTEL S.Mukhopadhyay 3/29

So, the instructional objectives are the following, first going through this lesson, one should be able to mention the distinguishing features of industrial automation computational task. A PLC basically is a computer, so it computes computation it performs computational tasks related to industrial automation. But, this task has certain very distinguishing features, compared to the other tasks, which are let us say done in an office environment.

So, we are going to you should be able to mention some of these features, mention major hardware components of a PLC. Describe major typical functional features and performance specifications for the CPU, Central Processing Unit, I O or Input, Output MMI, the Man Machine Interface and communication modules. Then, explain the advantages of a function module, so what are the advantages of a function module, you will be able to explain and describe some major typical function module types, so these are instructional objectives.

(Refer Slide Time: 55:29)



Indian Institute of Technology, Kharagpur

Processor Module: Software Features

- Counting : Software UP/DN counters (0-999) up to 10 – 100 KHZ
- Timing : Several Software Timers (ms-hours)
- PID Control : Different configurations
- Program Execution
  - Cyclic, Time Controlled, Interrupt Driven

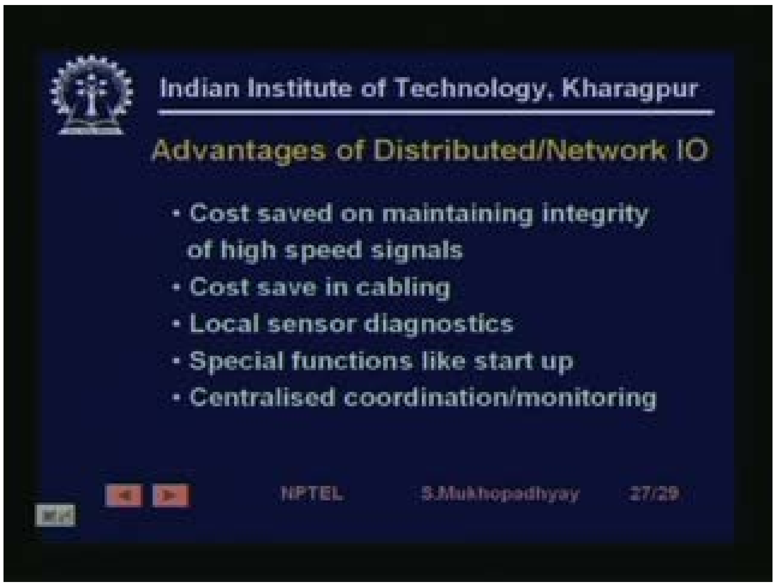
NPTEL S.Mukhopadhyay 10/29

The program execution in these systems is very interesting, there are generally three four different types of program execution mode specified. For example, a mode could be cyclic, by cyclic we mean that, you have a number of computational tasks, begin just like RLL program execution.

So, begin here come to the end start all over again, so this just goes on typically cyclic time remains more or less constant, but it could vary little bit depending on the program logic. For example, if you have some program control statements, if then else kind of statements; then whether some blocks will be executed or not will actually depend on the data.

So, program execution time is not always constant, it actually depends on data, but roughly it will be constant and in fact, it is it is preferable, that it is constant. So, that you are not surprised that, for some value of data, suddenly your programming execution takes a very long time and your deadlines are missed. So, it is preferred that real time programs are predictable and not too much varying time requirements.

(Refer Slide Time: 56:39)



Indian Institute of Technology, Kharagpur

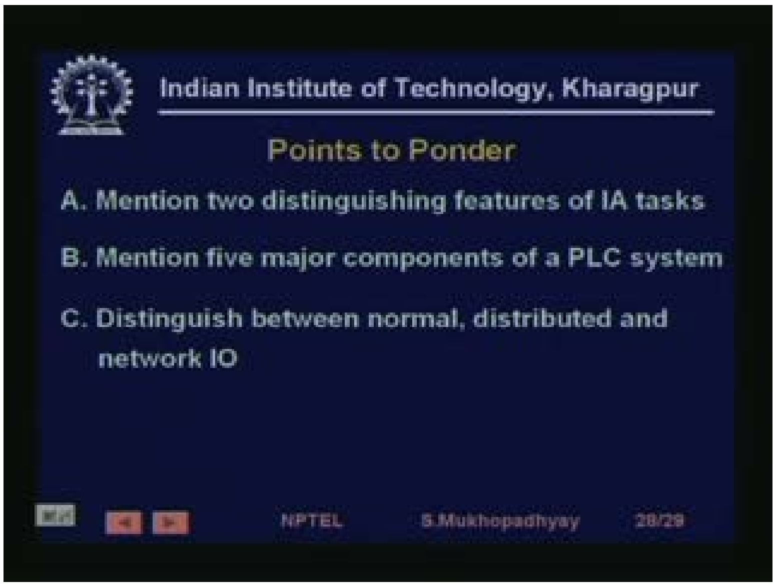
Advantages of Distributed/Network IO

- Cost saved on maintaining integrity of high speed signals
- Cost save in cabling
- Local sensor diagnostics
- Special functions like start up
- Centralised coordination/monitoring

NPTEL S.Mukhopadhyay 27/29

The advantages of distributed network IO are well understood, cost saving on maintaining integrity of high speed signals, because digital comes. Basically, the advantages of digital communication and the advantages of having an intelligent module near the machine. So, you can have good sensor diagnostics, fault can be much more, you know monitoring functions can be realized without overloading the CPU. You can do special functions likes like start up, so in a sense, in such cases the PLC, CPU, really works like a supervisory system and the actual controls is done on the spot, so you have better centralized coordination monitoring.

(Refer Slide Time: 57:17)



Indian Institute of Technology, Kharagpur

Points to Ponder

A. Mention two distinguishing features of IA tasks

B. Mention five major components of a PLC system

C. Distinguish between normal, distributed and network IO

NPTEL S.Mukhopadhyay 28/29

So, we have come to the end of the lecture and I hope you have got a fair idea, about what makes a PLC system and as is customary again, you have some points to ponder. So, think whether you can mention two distinguishing features of industrial automation task, compare to let us say a task in a bank, which are also computational tasks, which also communicate. Mention five major components of a PLC system, we have mentioned more than five, so you should be able to mention five and distinguish between normal distributed and network IO, so here we end today.

Thank you very much, we will meet again.