Digital Circuits and Systems
Prof. S. Srinivasan
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras
Lecture - 39
System Design using the Concept of Controllers

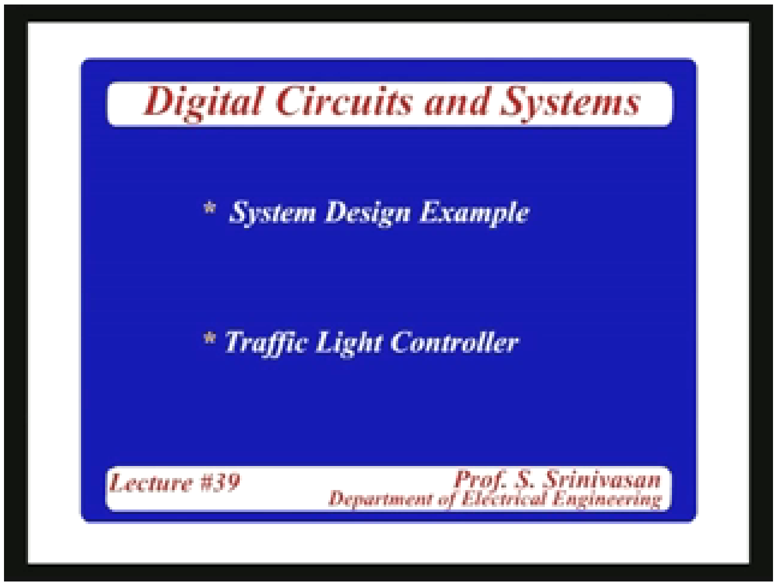
(Refer Slide Time: 00:01:05)

Digital Circuits and Systems

Overview of Lecture #38

Lecture #39 Prof. S. Srinivasan Department of Electrical Engineering

(Refer Slide Time: 00:01:18)

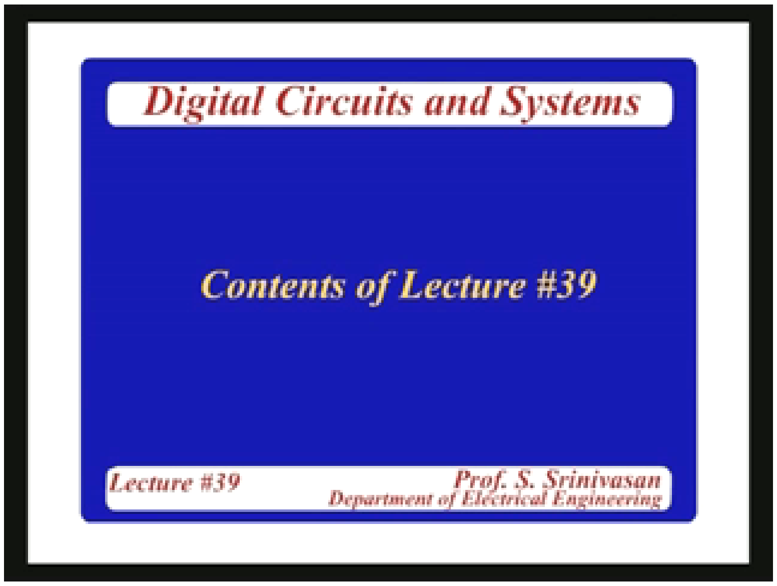

Digital Circuits and Systems

* System Design Example

* Traffic Light Controller

Lecture #39 Prof. S. Srinivasan Department of Electrical Engineering

(Refer Slide Time: 00:01:34)

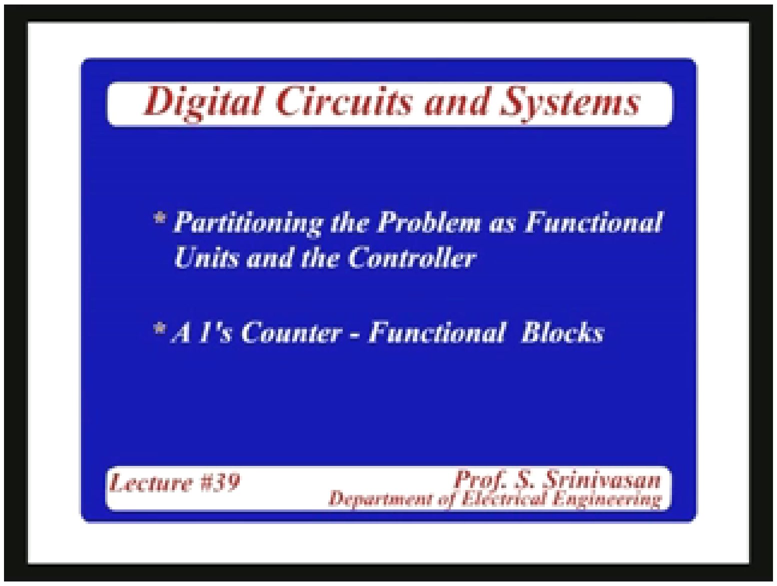


Digital Circuits and Systems

Contents of Lecture #39

Lecture #39 Prof. S. Srinivasan Department of Electrical Engineering

(Refer Slide Time: 00:01:43)



Digital Circuits and Systems

* Partitioning the Problem as Functional Units and the Controller

* A 1's Counter - Functional Blocks

Lecture #39 Prof. S. Srinivasan Department of Electrical Engineering

In the last couple of lectures I think we saw a design of a traffic light controller. The point I was emphasizing, even though the individual components of the design are all familiar to us from the earlier treatment of combinational sequential logic, the point I was trying to emphasize in their design example was to understand the problem clearly, specifications of the problem and identify the input output signals required to get a state graph which will satisfy all the requirements and once you have it your standard procedures are of implementing or translating this state graph into hardware.

I want to do one more example, emphasize another point a slightly different point here. There are two types of designs in the sense one is the control aspect. In the traffic light control the control is the most important thing, there is no hardware involved except lights and may be a timer. So you can incorporate the signals required for those into your state graph. On the other hand, there may be designs where you may have to design around components which are available these are called off-the-shelf components off-the-shelf elements like you are trying to do a Arithmetic Logic Unit for a computer, you don't go and design arithmetic circuit, subtractor circuit, adder circuit, multiplier circuit, logic circuit so these are all there you can buy a 4-bit ALU as they call it arithmetic logic is ALU unit 4-bit ALU and 8-bit ALU and then control this using the algorithm. I already defined the word algorithm to you I think more than once. That is the systematic procedure or the sequence through which the circuit has to go through in order to implement the given problem.

So the controller controls the functional blocks in accordance to the specifications of the design but in order to do that he needs to understand what are the signals available from these functional units. What are the signals available for controlling these functional units as inputs, what are the signals available from the functional units as outputs which can be incorporated into our state graph because our state graph actually has states namely the external inputs and external outputs, the state in which the circuit has to exist based on the sequence through which it has to go through to implement a given example and on the other hand, we also have inputs external inputs, a typical one input one output example we have been using x as the input and z as the output, the circuit can exist in several states so when x is 0 it goes to this state and when x is 1 it goes to the other state and so in one state z is 0 and in the other state z is 1 so all those things.

So same things now will be delivered given by those functional units which you design, the 4-bit ALU is an example. For example, how do you make a decision, how do you do an addition rather than a subtraction from a 4-bit ALU so what signal should be given to a 4-bit ALU for you to make an addition possible, what signal should be given to it to make a subtraction possible and so forth, the logic operations possible. So input signals of the ALU you have to come from your state graph which are the outputs of the state, the C of the state graph, the outputs of the state graph acts as the input to the functional units. Similarly, the outputs of the functional units like the 'operation completed', suppose you are doing an addition with a carry a overflow, if the carry is there you have to proceed and if the carry is not there you have to stop let as say a simple example, then I should be able to take this into account and feed this into my state graph, carry as the output of the functional block ALU but that becomes an input to my state graph.

If C is 0 carry is 0 I go to the next state C is 1 I stop. So the inputs of the state graph or the outputs of the ALU or the functional units and inputs of the functional units have to be derived from the outputs of the state graph. The output of the state graph becomes the input of the functional units; output of the functional unit becomes the input of the state graph so I should have a tie up between these two. Our design is based on the availability because every time when somebody asks you to do a small thing you cannot go from scratch, give me 15000 AND gates and I will design the circuit and give it to you, you can't say that, you have to work with what is available.

We already saw a flavor of that in multiplexer-based design, ROM based design, PLA based design and so forth. we have to carry that large circuit like ALUs 4-bit ALUs or even I may have a memory unit, how do you put things into the memory, how do you read from the memory, what is the timing signal required, what is control signal required for reading operations from memory, what is the control signal required for writing operations in memory, what are the timing signals I have to give etc and anything else for that matter.

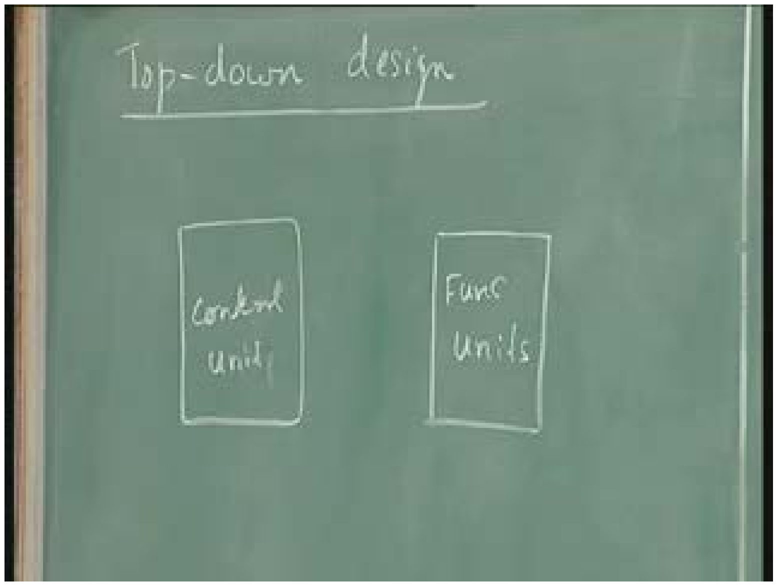
So the example that we took earlier the traffic light controller did not have many of those features because the functional units there were very simple. The functional units were the lights and lights do not have anything except, if you give a signal it glows and if you don't give a signal it doesn't glow. So we did not separately treat them as inputs of these functional units going to the state graph and the output of the state graph going to the input of the functional units so we did not see that but we integrated the whole thing into one design. But many times for most practical designs we have to separate out the functional units from the control units. This is the approach.

Again, I have used this word earlier, the top down design we call it, we have to have an overview of what we want to do, a calculated chip you are designing, what are the things that are going to be available already to you around which you are going to design. Of course finally you integrate the whole thing and bring out a chip for a calculator. It does not mean that the individual design have to go from gates, does not mean that individual design has to go from Exclusive OR gates for half adder, you can have a 4-bit full adder or 8-bit full adder which is available as an integrated circuit and design it around it and finally you can integrate the whole thing as another IC.

So the functional unit, control unit concept becomes an important part of a digital system design when the system becomes a more involved system than a simple controller like in a traffic light controller, elevator controller or a washing machine controller where you have to just detect the level of the water, stop the flow, these things are simple I can integrate, even there I can functionally partition it, I can functionally partition the traffic light controller, the time and these are the external elements that are available to me. The timer requests a signal called time on or time start and the timer gives an output saying the time is over so I can treat this as a functional unit time. It is the functional unit, timing is the input, start time is the input and end time is the output and this input goes from the state graph and output goes back to the state graph. I can treat it that way but I don't have to, this is a trivial design and I don't do it. But when the design becomes complex, a microprocessor, a computer, a microcontroller and so many other automotive electronics many features are to be controlled where functional units are available or defined already by somebody else either they are available as standard commercial components in which case you call them off-the-shelf components.

What you mean by off-the-shelf components is you can go and buy them from an electronics store; off-the-shelf they take it and give to you, called the off-the-shelf components. Or somebody might have developed it, it may not be off-the-shelf component but somebody might have developed it for some other application we do not have to redo the design, you might have done it, your company might have done it for another design, plug it out from that fellow and put it in your design as long as there is no copyright violation you can do that. This is what I want to teach you; the partitioning. It is another approach to top down design. Top down design works in two levels; the level we have been talking about is define the problem, complete overview of the problem, give complete specifications, input output signals, find out what you need, draw the state graph, from the state graph we go to implementation that is the top down. but here I am going one level above that we have a overview of what it is, what are the functional blocks required, which functional inputs are already available to you, which functional units have to be defined by you and how are they interconnected in terms of relations of timing and signals and then try to drive the state graph. So this is the top down design in its higher level or the purest form. so what I am going to do is, any problem will have a functional unit and control unit. it may be trivial or it may be complex but it is possible to partition any digital system into a functional unit and control unit, the control unit is one but the functional units may be many.

(Refer Slide Time: 12:28)

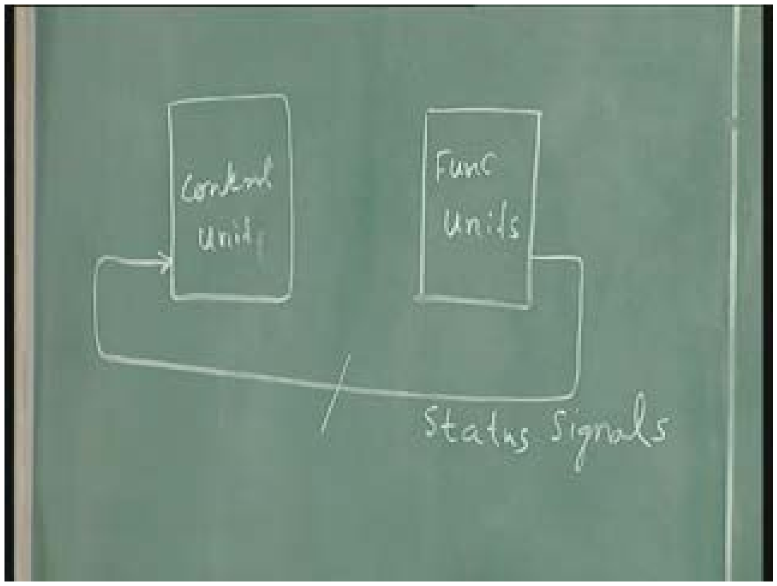


If it is trivial you don’t do all this paraphernalia, if it is non-trivial you do this paraphernalia. So the functional unit’s output has to be monitored before….. I can’t say, suppose I want to do a multiplication by repeated addition, what is known as the shift and add algorithm I shift the digit and that is what we do normally by paper pencil. I can add it, before adding it I should shift it.

Why do I need this feedback from functional unit back into the control unit? I cannot start a step in the control without knowing the results, the previous step or when the previous step is not completed. So I need the information from the functional unit back into my control unit. These signals are called status signals. The status of the functional units has to be known to the control unit.

The one line does not mean one signal, I am just giving you a……… in digital parlance, in digital convention they put a slash, when they put a slash across the line it means there will be many signals or they put a thick arrow, you might have seen this in books. The thick arrow means several signals or signal slash. If you know the number you put the number here and if you don’t know you just leave it.

(Refer Slide Time: 14:02)

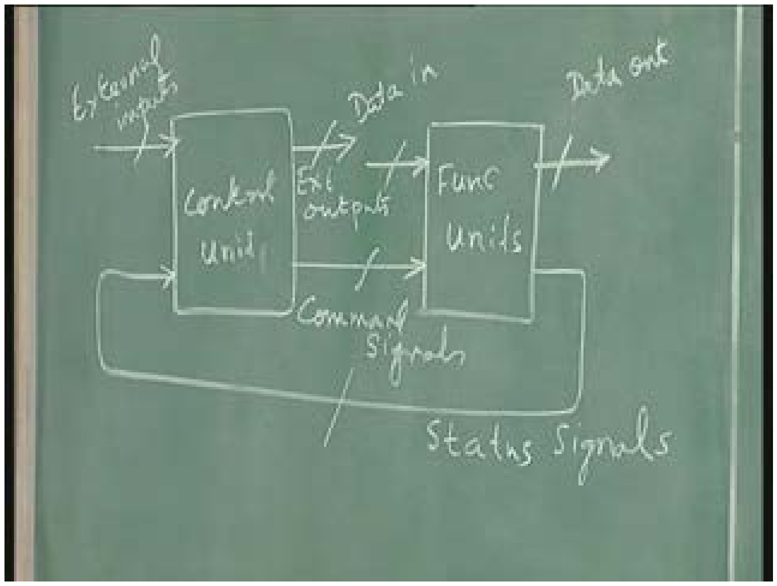


So, several status signals are required by the controller in order to design the next thing to do. Likewise based on this the controller issues the signals which are the command to the functional unit, do it. Now you have finished shifting now go and add or you finished addition now go and multiply. So the next step is always given out by the control unit to the functional unit so this signal naturally should be called what? If this is called status signal then what will be the logical name of this signal? I find the status and give command. I said I give command. The controller gives command based on the status that means this signal should be called a command signal.

But the functional units do not operate on these signals. The functional units are only controlled by these signals; the control signal and the command signals and status signal are the status of it but they operate on data. I cannot add numbers from the control signal, the numbers have to be fed separately. So, in addition to this I need to put in data in data in and data out. So data in general form, data does not always mean it is numbers, if it is a multiplication it is data, addition is data, storing is data or some other operation it is some signal. Some signal has to be processed let us say. This functional unit processes the data, puts in the output, and in that process it cannot do it arbitrarily at its own will and pleasure, it will do it at the behest of the control command signal and gives its status periodically back to the [con….16:07] saving that I have done it so what's next I should do. Likewise I can also have control unit's external signals because starting the whole operation need not come from the functional unit, it may not be a status signal. I may have to give an external push, a push button, let's start a reset, when the computer hangs what you do? You push the button reset button. So sometimes you have to restart, sometimes you have to start from the beginning, sometimes you have to give some external signals which are not related to the functional operation of the functional units.

So, in addition to these command signal the status signal inputs and command signal outputs and this control unit also will have signals called external inputs typically as I said start and also periodically gives…. suppose the operation is completed the result is ready so you can go and look at the result that also can be a function of the control unit. The control unit, in addition to giving the commands can also give out signals to the external world saying that the operation is completed or hanging error, I cannot proceed because this data is missing or in compatible, whatever you want to display. It need not be the result but some sort of an output for you to know what is happening. So, in addition to control, similar to the external inputs you can also have external outputs.

(Refer Slide Time: 17:58)



This is just a generalized model of a top down design of a digital system. Now within this there is no rigidity in this partition. The timer could be outside or timer could be inside. If you can count every clock in a control loop then the timer can be inside my control unit. Or if you put an external counter to count my time then it will be a functional block which can count so there is no rigidity in the partitioning because there are certain things which are specifically a functional unit like ALU as I said as an example. I cannot think of ALUs as part of a control unit but if you want to you can always do it but. You can push in things into this or from here to here. so, partitioning is…… it is not a rigid partitioning, or what shall I say, this is a soft partition, you can move things around like some of those modules they put in office space nowadays, you know they just remove and then….. In conference halls you might have seen this. There is an open session, everybody will be there and then they will remove all the in between partitions and then suddenly they will break into three different sessions, suddenly a wall would have come up when you have gone out for tea, and when you come back it will be three different rooms. You might have seen this I don't know. When you start attending conferences you will.

Now what we have been doing till now is of course, functional units separately we studied, many functional units in this course; counters, shift registers, we did adder and subtractor, we did even multiplier I think, multiplier example I remember doing and we have done other things; code converters and things like that. In control unit what we have been studying in the sequential circuit design, state description of a problem, implementation of the state and if there is no specific functional units as a required thing we put into this control unit every thing, I have even called it at that time control unit we call it full design because the system is such a simple design, we did not partition it into control unit and that is what is the extreme of this soft partitioning. The entire control unit can be gobbled up I mean entire function can be gobbled up by the control unit, of course the reverse is not true. As long as the sequential machine it has to go through states and states are represented by control units.

So we have been concentrating on this and the example we did last time. the example we did last time we concentrated only on this (Refer Slide Time: 20:57) the external inputs with the timer signals, external output with the lights, there is no command or no status because there is no functional units so we use this internal inputs and the states were defined in this and it went through different states. And this further has to be very clear specified. Everything has to be clearly specified. If you are vague your design will be vague and if your design is vague it doesn't work properly and if it doesn't work properly then you are not satisfied, it malfunctions. How critical it is depends on the operations.

Supposing it is some TV remote which doesn't work you do not worry, all is you have to get up from your sofa and go and do it manually but then if the traffic light fails there is chaos and if a flight control fails you know what the result is. So these are all different things, level of criticality. So everything has to be defined clearly. Once a problem is given to you, you should know whether there are enough or known functional units which can implement this, list them out and after listing them out to find out what is the input requirement each one of this or what are the outputs available for each one of this. Inputs of these have to come from here outputs of this have to go back into this. This is addition to data. Data input and data output is independent.

Supposing I have an adder the data has to be there. A 4-bit adder means there should be two 4-bit numbers, the result will be a 4-bit number with a carry 5-bit number. In addition to that is there any start signal available, is there an enable signal available, is there any end of addition available, so for each functional unit you should list down the control signals available as outputs and control signal required as inputs, when does it start. For example, you take a shift register, the load is a control signal, clear is a control signal so these are the things you should list out and then based on that you go to the…… start your sequence. Available functional units and required signals are listed based on that you start your state graph.

What you have been doing till now is to ignore the first part because we have been taking very simple components and standard components and we were only starting with the state graph. I am now taking you one level up that's all. There is no contradiction to what I am saying so we can do a simple design of a counter, the counter is available so what is the design, you can buy it and do it otherwise we can get flip-flops and connect them together like that, that is what I am saying.

At different levels you have to go on because finally it is……. today's digital world is so complex in terms of the number of functional units and the complexity of control, functionally they have not become very complex, whether it is a Pentium processor or a simple calculator or a 80 85 microprocessor, addition is addition, may be there are some techniques which make the additions little faster in Pentium processor compared to the olden days. Complexity does not come in the functions, complexity comes in the……… the control has to be……….because so many things are there to be controlled at the same time and timing has to be very strictly maintained.

So, after understanding the problem clearly partition the problem into functional units and control unit. Functional unit, you list them and see whether they are available and if they are available see what are the data input data output requirement plus what are the control input requirements and status output available and start your control unit by means of a state graph.

So our state graph implementation of the digital circuits have been systems, sub-systems we have been discussing so far is only this part. because of the soft partitioning some of these things we have been able to push like the traffic light controller part where the light was inside this and some other things we did earlier; sequence detector you remember we did some and for those things we did not talk like this, we just put it all together. So this comes the same way, there is no change in that procedure, there you have to draw a systematic state graph starting with the various sequence of events the circuit has to go through in the case of a traffic light controller, we said the sequence of the lights has to go through, we drew diagrams to illustrate them and then do the state graph and from state graph…… Every time in the state graph your comparability must be maintained between the functional unit and the state graph.

Any input required in the state graph has to be available as a status signal for the functional unit. Any output coming out of the state graph should be able to go as a command signal into the functional unit. That compatibility has to be maintained when you draw the state graph. Sometimes it may not happen. Once you assume that something is possible it requires an extra signal control signals, we do not have it, how do you do it, you modify your functional unit's requirement.

Your functional unit is an open world with electronic shop as I said to go and buy. If something doesn't suit you, you throw it out and buy something else. I will give you a simple example. Supposing I say add, shift and add I say as an example of multiplier. I give a shift signal you can shift, a shift register, add. We have seen 4-bit adders or 1-bit adder we saw, we also saw half bit….. half adder. For a 4-bit adder we give four inputs, 4-bit input for A, 4-bit input for B get some 4-bits and carry 5-bit output. As well as A and B was given the sum was available based on the timing delay of the full adder. Is there any control signal as input which says add in the adder? Did we see that when we drew the full adder using Exclusive OR gates and all those things, even carry require adder we did, and the full adder is a combinational circuit, I have been maintaining this, it is not a sequential circuit unless it is a serial adder a 1-bit adder which we are not talking about here, we are talking about parallel adders.

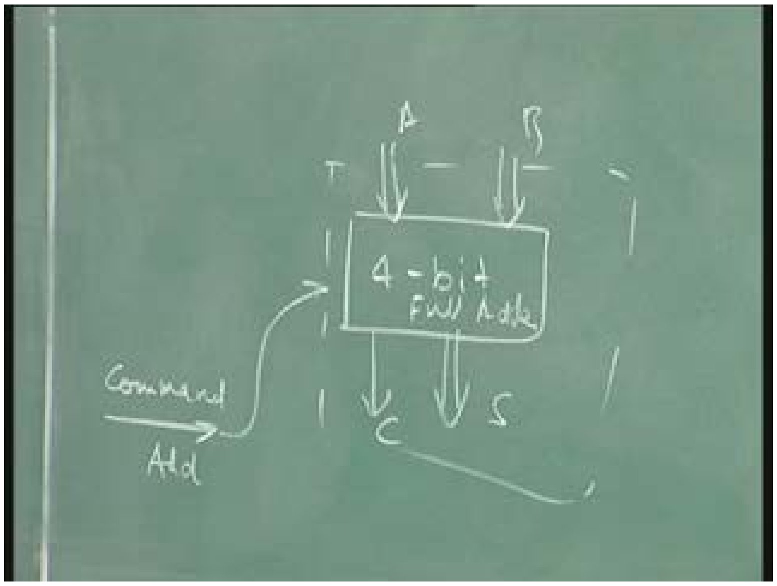
When you say 4-bit full adder you mean that four bits are given as A and four bits are given as B and some four bits and carry one bit comes, the five bit comes as the output assuming ideal components immediately, practical components after delay depending on the gate structure. Supposing a control signal says add and shift, control add comes from the controller. When we draw the state graph for multiplier I will say shift and add, it is a shift and add algorithm. So, shifting is no problem, there is a shift control in a shift register, if it is not given it will not shift. But for add, in a 4-bit adder is there a control called add? Tell me, so how do I use these signals, sometimes how do I use this control, how do I control my adder, how do control the adder when

I want to add? As long as input A and B are there the output is there immediately and here we are talking about clocks which are slower and those combinational delays are very small, nano seconds delay.

Combinational delays are nano seconds whereas clocks depending on the frequency operation will be of much larger periods so it is not good wait for you to given an add signal, as soon as A and B are given it is going to give an output, so now I want to control it when I want to add, I can only add after shift or shift after I add, whatever so some sort of a control is required, then I think okay may be a 4-bit full adder is alright but some other thing is there. So 4-bit adder can feed you into a register the output register can be controlled, I can load it when I want to add it.

Suppose I have a 4-bit adder, please understand what I am saying. Now, this is A, this is B and this (Refer Slide Time: 29:29) gives me C or sum and this be carry. Now there is no other control in this except power supply and ground. now I have……. coming out from the command signal controller called add, this has to go into this only when this comes this should add these two numbers otherwise it should, not possible, it is not possible, I can't just control somebody combinational logic, the moment I think the output comes, so how can I control it.

(Refer Slide Time: 29:59)



So what I do is, as a simple solution I put a register, after all this (Refer Slide Time: 30:12) can go into a register, adder, output can go into a register, how many bits five bits because this is 4 plus 1 so 5 bits so you get a 5-bit register and the register will have a load signal and I will give this command into this load signal. So even though I am…… I am having the illusion of controlling the adding, I am not controlling the adding but I am controlling the loading of the adding but the result is available at my request, at my command. At the controller's command the adder result is available. Adding must have happened earlier but I am not going to use it, ok you do what you want but I will look at you only when I am free, that is what the controller says to the full adder, they don't speak anyway but I am just……..

Why am I saying this? So when I draw the controller, state graph some of these nitty-gritty have to be sorted out so I cannot simply blindly put a signal a command signal called add, command signal called control, command signal shift and you have full adder you go and by the full adder from the market and come and then there is a hanging signal you have wired up this part, you have wired up that part, there is a command, p add command, I mean add command you don't know where to insert it in the bread board, you should have thought of it in the beginning. So I looked at the….. so that is what I am saying, when I look at the functional unit, in addition to data input data output I list out all the control signal requirements and the status signal output that is available so I will see that there is a missing command signal. So missing signal called add, so you go around asking the entire electronics market do you have a signal adder which will add, may be there is one with an enable, so something like that you by, otherwise your solution is so simple, get one more register in addition to adder and get the result of the adder into the register and control it.

A nitty-gritty is like this, have to be sorted out when you draw the control unit and functional unit minor changes, you cannot run one more time to the electronic store so you have to finish it before coming out of the inventory list. Likewise in the command signal also you try to accommodate everything, it is not possible, then I change my controller a little bit. Ok you tell the customer this is not possible, I can do it this way if you want. You will say yes or no but if you say yes, proceed.

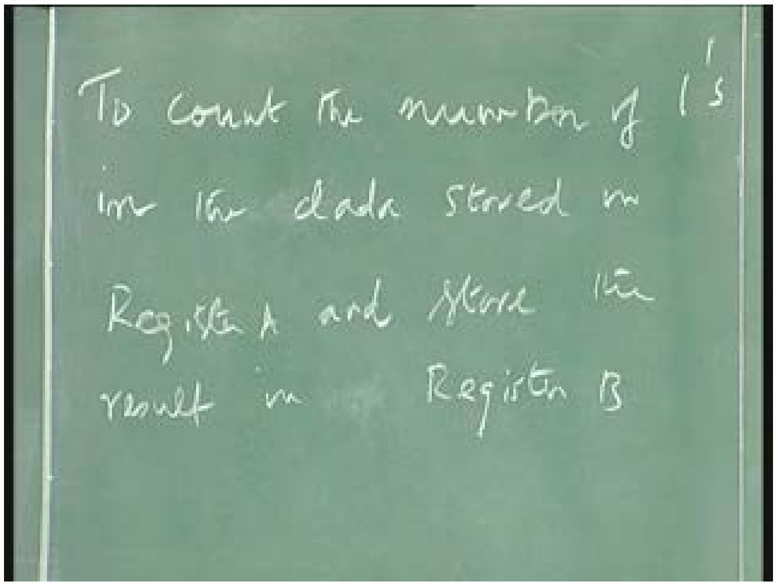
So I may have to modify my control unit and the functional unit slightly, fine tune it, trim it, based on the availability of the functional units, based on the requirement of the controller. Once I am done rest is easy. I don't have to teach you how to do a functional unit, there is a command unit as a…. control unit as a state graph. Once that stage is completed it is sealed from then on it is a state graph, transition table, flip-flops or a multiplexer based, ROM based, PLA based. Again and again I am emphasizing this fact, I might have told you this hundred million times in this course.

Design is the most important part of digital. Putting it together, of course you should know what is available, you should know how to do it most efficiently, most reliable component, you should know where the power supply should be given, but what is the most important part of the engineer's job, design it properly so that it is reliable and it works to the exact specifications given to you because we don't know how critical the application is. We can't be vague about it. If it is vague you ask for the details and if something cannot be accommodated go and say this is not accommodated, what else we want in this place and you give the alternatives, don't go to the customer and say I can't do it tell me some other way, no, you are supposed to give the alternatives. I can do it this way, this way, this way choose but not this way you want, these are the other three alternatives, choose one of them. So this is how a full real digital system design is done. So this will be the last example in this course.

I am going to take a simple example to illustrate this concept. The example is so simple it is not going to take much time. But I wanted to introduce this major concept. Finally I wanted to go from the definition of a gate and reduction of gates through various Boolean algebra, Karnaugh Map, combinational design, flip-flops, sequential design and then the MSI components, LSI components, state graphs, through system design some of the systems they did but I thought this will be the….. I will have to go to this level so logical completion of this course so I will have to take one more example which will illustrate this point.

Let us define this; this is a very simple example as I said. Again I cannot design a microprocessor or a calculator. I cannot design a computer in a class like this but it is the beginning of these things. The example is a very typical example. I have a register…… many times you want to do some operation, manipulation, you want to find the number of ones, odd bit, even bit, we talked about parity, remember long ago, parity is what, the number of ones in a bit stream, is odd or even? If it is odd it is called odd parity, if it is even it is called even parity. So, suppose I wanted to design a parity generator for that I need to do some counting of the number of ones. Suppose I have a bit stream and I just want to do a simple job of counting the number of ones in that bit sequence that's all I need to do, this will become a part of a bigger unit because I am designing a computer in which several things are there and a small local unit is a circuit to count the number of 1's in a particular register. So I have register called A, the problem is to count the number of 1's in the 8-bit data…… not even necessarily 8-bit, the data stored in register A and store the result that is how many 1's are there in register B, very simple job.

(Refer Slide Time: 37:37)

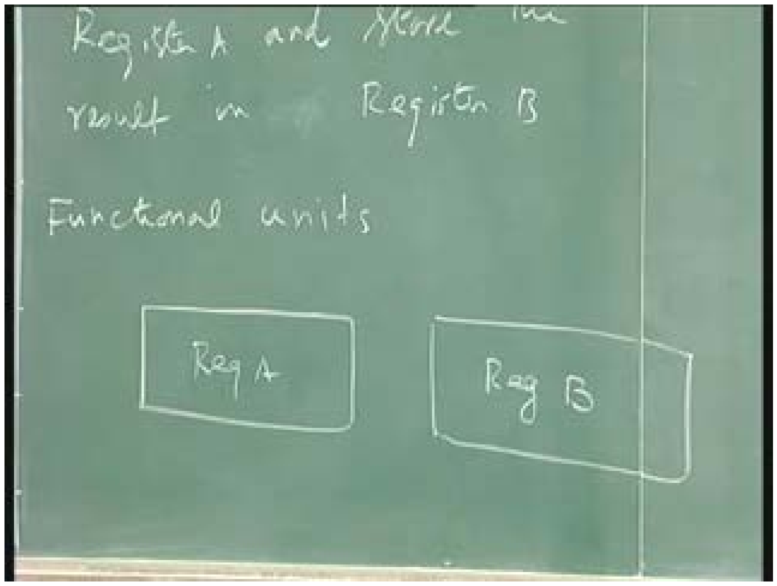


To count the number of 1's in the data stored in Register A and store the result in Register B

I have two registers: the register A and register B. Some data which is available I will first have to store it in register A, count the number of 1's in that data and then put that number (the number is a number of 1's) in another register called as B that's all. I can possibly do it without a functional unit partitioning but may be difficult but functional unit partition makes problems so simple. Just to give you an example I thought of this, a very simple example if not (38:15) pitched on this problem as a simple problem for you to understand the concept of partitioning between functional units.

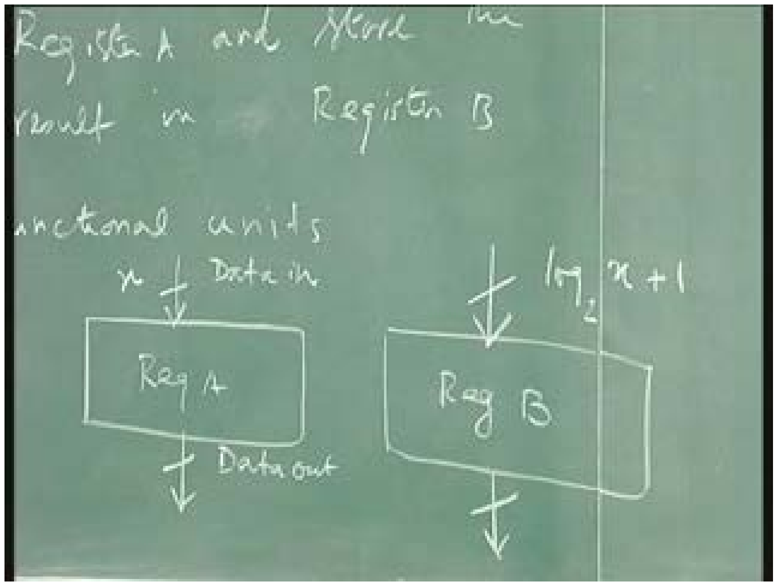
So functional units, what are the functional units here? Register A and register B. You should also know the control signals required for function A and function B and the output signals available from. Then I will have to do a state graph of the sequence of events, how to load the thing, how am I going to count it and when the result comes how am I going to write it into register B so all that has to be put in the part of a control unit state graph. Once the control unit state graph is there I will have to implement it using multiplexer based solution or whatever is the most favorite solution of yours.

(Refer Slide Time: 39:27)



So, first part is, as I said, the functional units. So I will call this register A and register B. This is not complete. The functional unit has to completely define the registers. Number of bits, command signals, control signals required for this to operate and status signals has come out as the register as the output can be used in my state graph. So number of bits I don't know since the problem did not specify the number of bits to be stored that will be assumed that the customer will tell you that so if it is a 8-bit register he will give you eight bits data will be stored you get an 8-bit register, 16-bit data is to be stored you put sixteen bits. So now I will use n-bits. This is data in (Refer Slide Time: 40:15) so the n-bits are available also. So I need a parallel-in parallel-out register which can hold n-bits where n will be defined by the customer, client and what should be the size of this register B? There are n-bits, how many 1's can be there? n 1's. So n bit register…. so that can be how many, what is the size of…..? Say it again, logarithm, log of n base two. So the size of this (Refer Slide Time: 41:19) should be… ok so n-bit….. log of 2 x plus 1. Supposing all the eight are 1's 1 0 0 0 so 4, I need one extra bit. So if there are eight bit number all the eight or 1's…… I will have to store how much, eight, eight is what? 1 0 0 0 so I need a 4-bit register, if it is seven it will be a 3-bit register so it is log of n plus 1, this is also input and output.

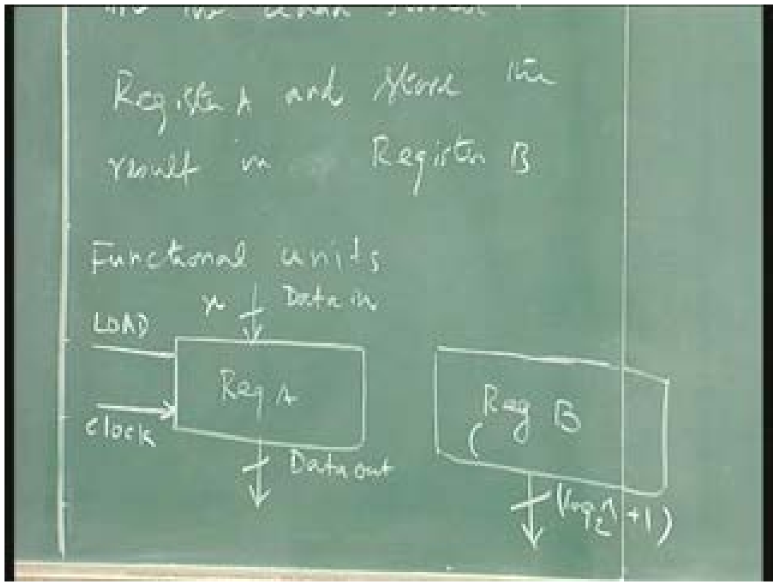
(Refer Slide Time: 41:59)



Of course, this need not be put there (Refer Slide Time: 42:09) I am just saying the size of this, this also can be parallel but we don't need it, this is going to increment by………so the size of this should be, this output is required, log of n because I need to use this in the next step of my computation, next step of my computation. So this will be parallel-in and parallel-out, this should be parallel-out as parallel-in is not required because I am going to count it. But then register B should have……… we will complete the requirements of A and B completely. Now look at A, A is a register which will store the data so it should have a loading feature, should be clock because only then it will load and clock.

I will assume that all of them are synchronous operations to avoid confusion. Loading, clearing and all the operations will make it synchronous. You know what I meant by synchronous? It is along with the next clock edge. As soon as the signal is given it does not do the job it waits for the next clock edge and does the job. So synchronous loading; when I want to load a new value I put the value here and put one load on 1 and clock it next time it loads.

(Refer Slide Time: 43:40)

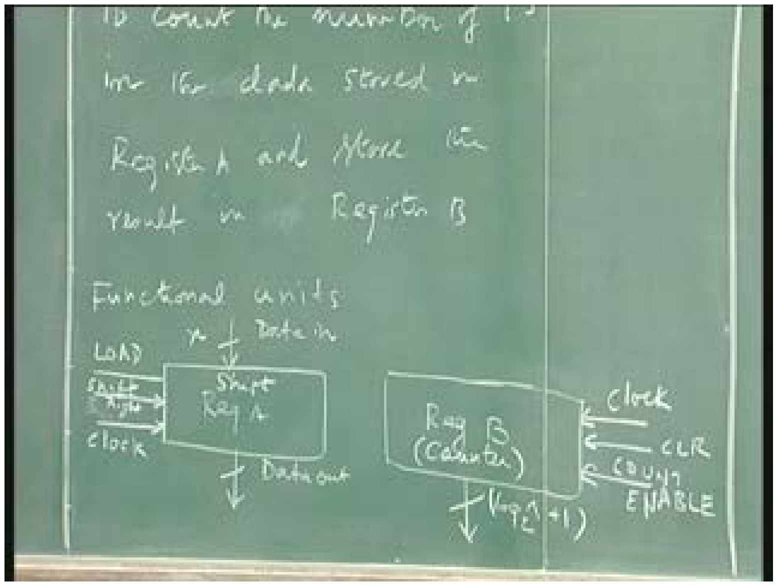


Is there any other signal required there? Now we have to decide how we are going to do that. So, algorithm has to be decided. How am I going to find out the number of 1's in a bit stream. In n-bit register in 8-bit register or 8-bit of data stored in register A how am I going to find out how many 1's are there.

We can do this in so many ways. I can add the bits and do what. What I can do is to look at one bit at a time and keep shifting it. So I will always look at the LSB Least Significant Bit. If it is 1 I will add 1 to the register B and then shift it because algorithm has to be simple. I can think of so many others ways of doing it. but the simplest way I can think of is or one of the simplest ways I can think of is look at the last bit LSB if it is 1 add 1 to my B register shift it by one bit again look at it and keep a count till n is over and when n is over stop the process. That is going to be my control algorithm; the state graph will be decided by that.

So you have a rough idea of what I am going to do. I will go back; I will assume this, go start my control algorithm and if there is any problem I will come back and correct it, as I said correct fine tuning, back and forth going couple of times to trim the requirements. So right now I am going to say the shifting feature is required and I want to shift left shift right so that LSB will always be so I will say shift right. So my requirement of register A is an n-bit shift right feature and clock. And register B should be again clock and whatever you want to do, I will use as the counter, I will clear the counter to start with so I am going to use it as a counter. It will be a counter which will be cleared and every time there is 1 in the shift register I will count it. So I need a signal to count enable, to solve.

(Refer Slide Time: 46:36)

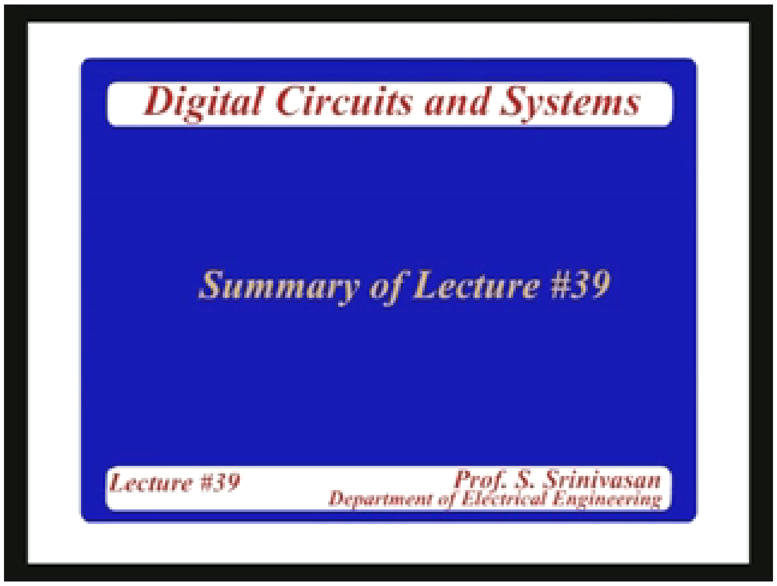


So, as of now I am defining two registers, its architecture and functional units. one register should be a shift register in which I will load the number, look at the LSB, if it is 1 I will add 1e to the register and if it is not I will shift it and again look at it and keep doing it till all the bits are covered, how I am going to do that we have to think about it, it will all come in the algorithm.

But now the first cut, the first cut requirement I have to give now so these are the specifications I am giving. I may have to change it a little bit. So right now I am saying a register which has a shifting possibility and loading possibility whose data-in and data-out are available is what I need for A and for B I need a counter which can be clear to start with and enabled whenever we want at the edge of a clock and then the output should be read, the parallel output and finally I may insert it.

How am I going to use it, I will start my state graph and then when the nitty-gritty of the signals are available, if something is not available I have to go back and make a little modification of this. But so far whatever is assumed is available in the market, it is off-the-shelf. I can buy a shift register with loading feature and shifting feature. Or I can by a counter which can be a clearing feature and count up feature. So I have not violated any common………. even if it is, it is still worth it because as I said similar requirements somebody they might have done it earlier and a design is available with you in your company and you can use it later. So we will continue this discussion and complete the problem.

(Refer Slide Time: 00:48:17)

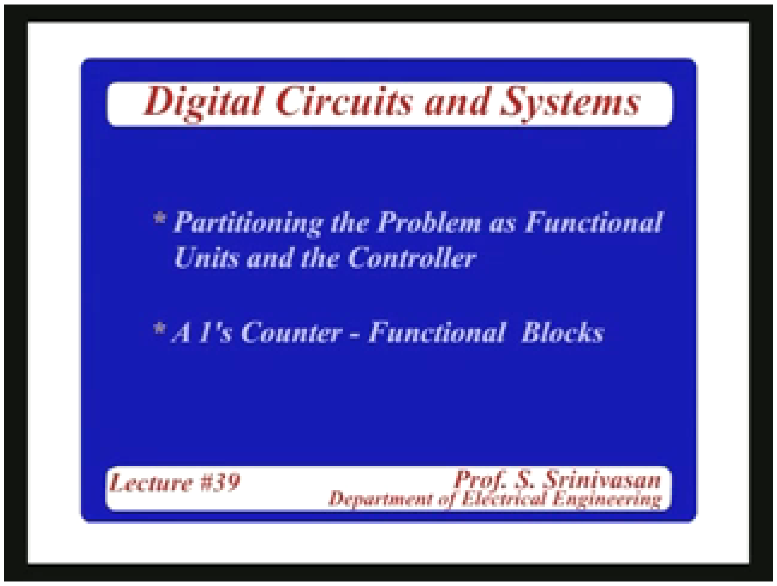


Digital Circuits and Systems

Summary of Lecture #39

Lecture #39 *Prof. S. Srinivasan* *Department of Electrical Engineering*

(Refer Slide Time: 00:48:27)



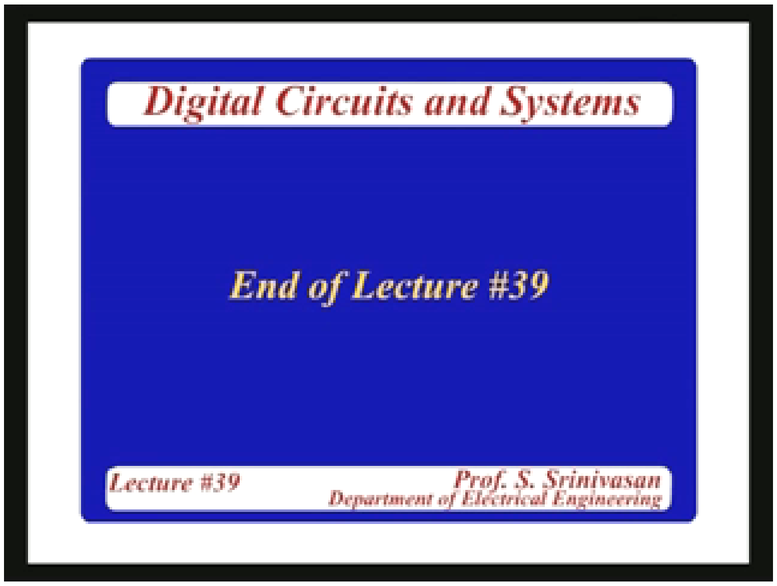
Digital Circuits and Systems

* *Partitioning the Problem as Functional Units and the Controller*

* *A 1's Counter - Functional Blocks*

Lecture #39 *Prof. S. Srinivasan* *Department of Electrical Engineering*

(Refer Slide Time: 00:48:40)



Digital Circuits and Systems

End of Lecture #39

Lecture #39 Prof. S. Srinivasan Department of Electrical Engineering

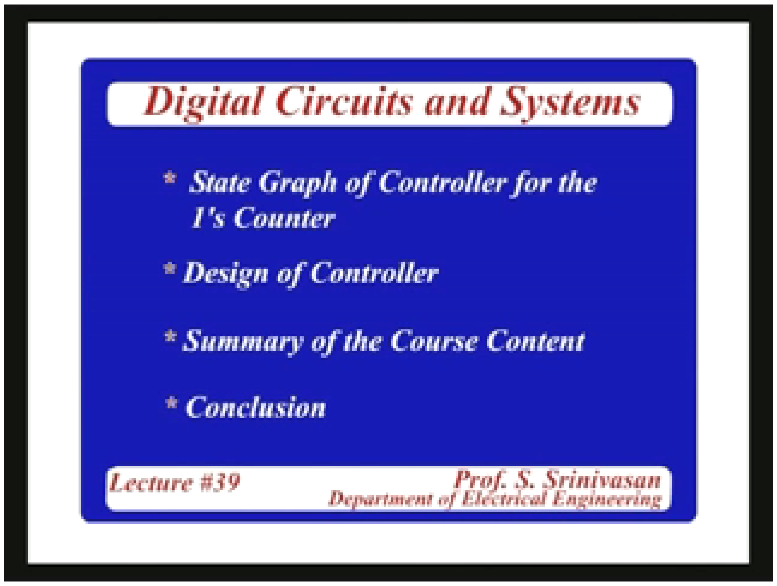
(Refer Slide Time: 00:48:46)

Digital Circuits and Systems

Next Lecture

Lecture #39 Prof. S. Srinivasan Department of Electrical Engineering

(Refer Slide Time: 00:49:03)



Digital Circuits and Systems

- State Graph of Controller for the 1's Counter
- Design of Controller
- Summary of the Course Content
- Conclusion

Lecture #39

Prof. S. Srinivasan
Department of Electrical Engineering