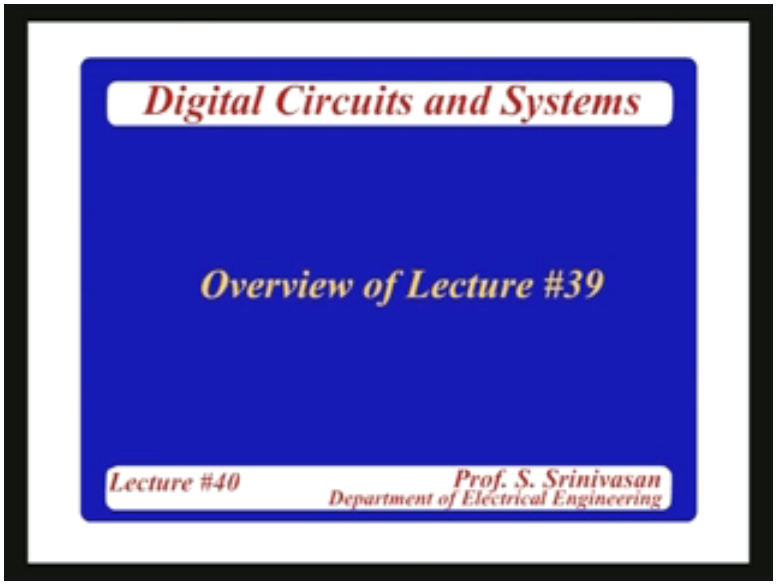
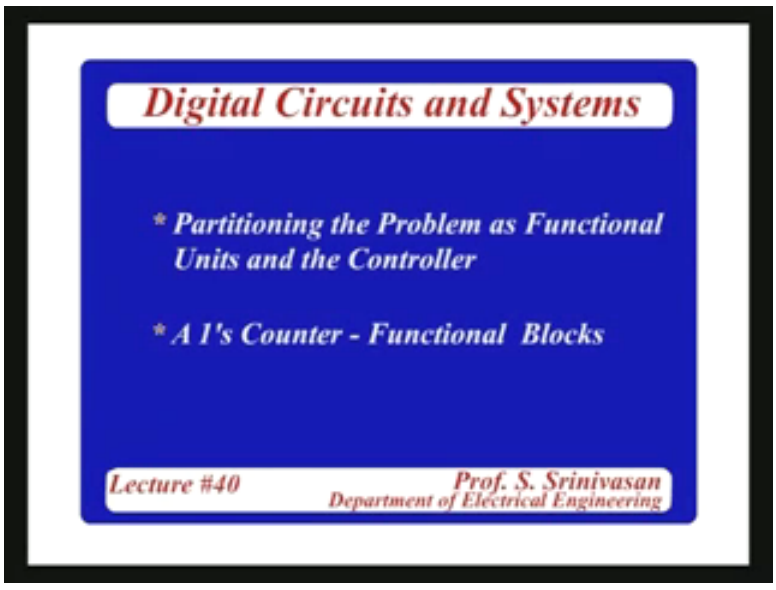
Digital Circuits and Systems
Prof. S. Srinivasan
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras
Lecture - 40
System Design using the Concept of Controllers (Contd...)

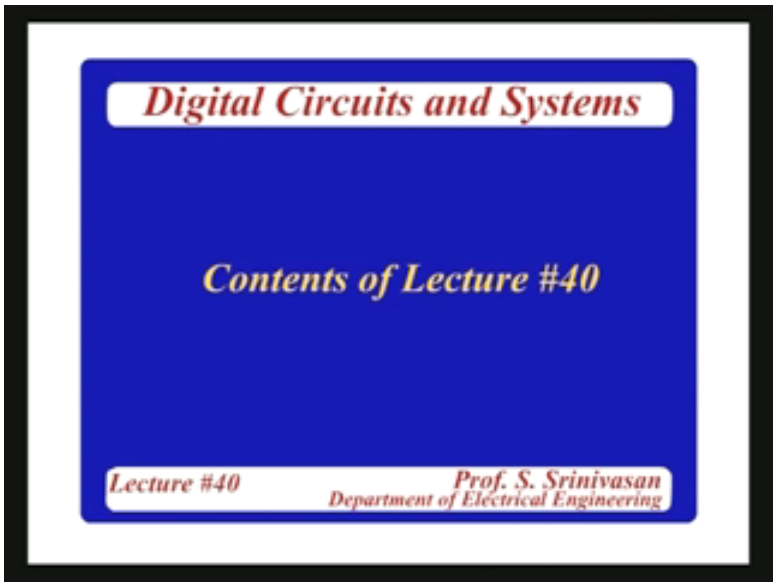
(Refer Slide Time: 00:01:07)



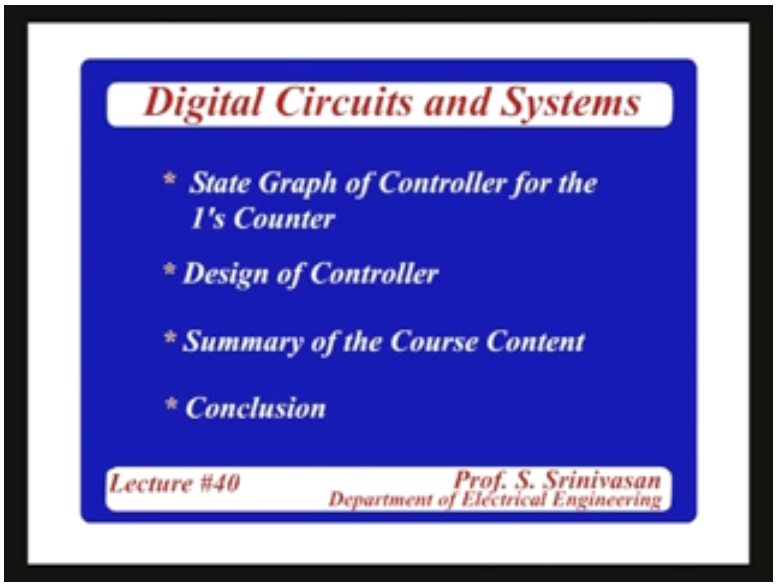
(Refer Slide Time: 00:01:12)



(Refer Slide Time: 00:01:23)



(Refer Slide Time: 00:01:35)



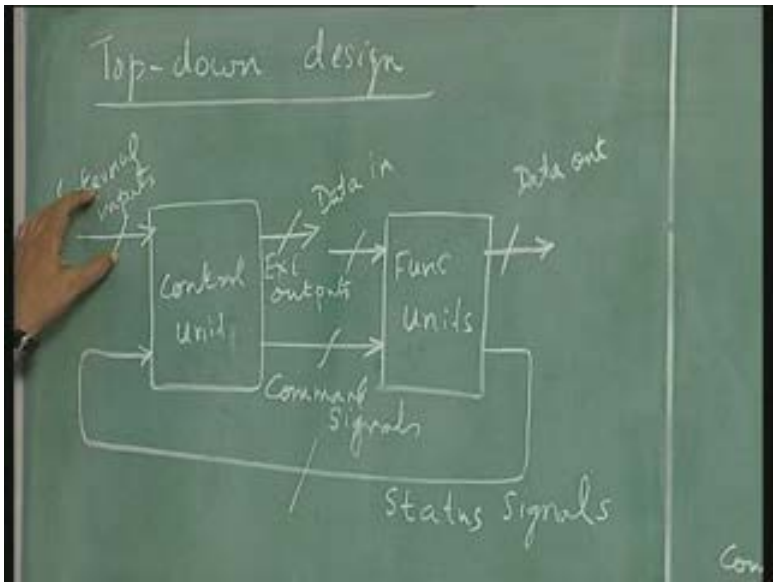
(Refer Slide Time: 00:01:53)

We were discussing a top down design example. Just to illustrate the concept we took a simple design wherein a register will load the data and we will count the number of 1's in the data and put that value of the number of 1's in another register and stop the operation. This is a simple system.

We talked about the functional unit requirements; we need two registers; register A and register….this is the partitioning between functional units and the control units (Refer Slide

Time: 2:40) data goes into the functional unit and data goes out of the functional units so the functional units are controlled by command signals, the output of the functional unit, the conditions of the outputs of the functional units are given back like a status information input machine which helps the control unit to steer it from state to state.

(Refer Slide Time: 3:02)



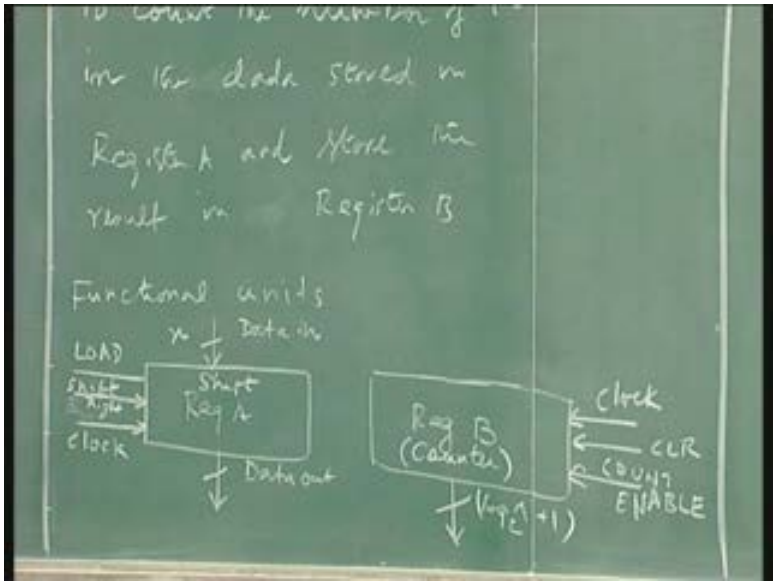
In addition, the control units can have external inputs and external outputs meaning things like start, reset and stuff like that. External outputs could be typically the status like operation over, complete or some other problem, error message whatever you want in this place, control outputs. These are the functional units (Refer Slide Time: 3:29). People call these functional units as architecture of the problem. I was avoiding this term but I thought it is better to give it to you.

Architecture is what is the concept of all these together. Architecture is the combination of the functional units, a group of functional units called architecture. So the architecture consists of two registers and the architecture is controlled by the control unit. So the partitioning of the control unit and functional units, some people call this functional unit part as architecture part.

Architecture is the one which processes the data, and the control unit is the one which controls the operation of the architecture. I have used the word functional unit because it is more sort of clear to understand, easy to understand. the architecture or the functional units whatever way you want to call it, this particular problem consists of two registers; register A which will load the data, since we do not know how many bits I am going to put n-bit register and it should be a parallel-in so that we can load the data and parallel-out if you want to take it out and then if we want to load it, when we are ready to put the data into the register we need to load it so there is a load signal required, all of them are controlled by system clock so there will be a clock. Then one feature, the algorithm we are going to do is, there are several ways in which we can do this problem.

One of the way is to look at the LSB the Least Significant Bit of the word and see if it is a 1 or 0 so if it is 1 then add 1 to the register B otherwise go and shift it and look at the….., you always look at the LSB but shifting it till you exhaust all the bits. So we need a shift register with a shift right feature. A shift right feature, loading feature, a clock, data-in and data-out etc are the specifications or the requirements of the register A.

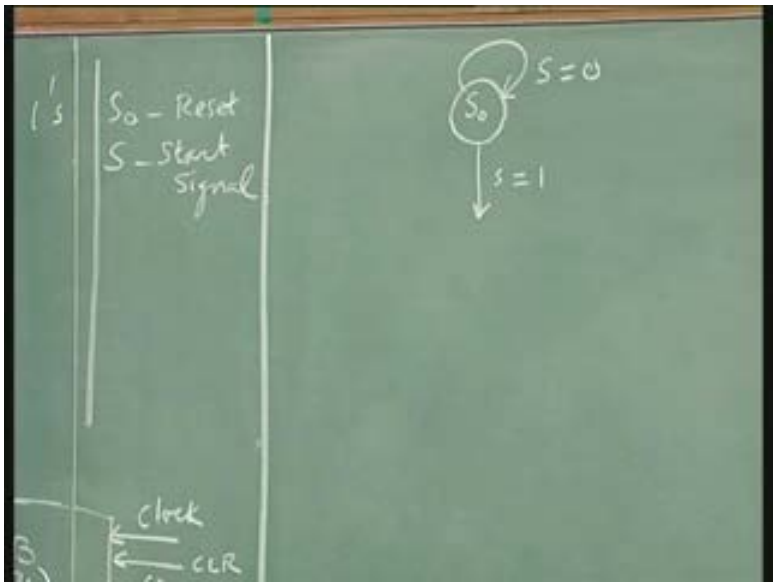
(Refer Slide Time: 5:31)



As I said earlier, you are supposed to have a component which is available either commercially or with you in your earlier design as a code or an IP they call it, IP stands for intellectual property. That is, even in an IC design even though you build from gates, supposing you have a particular complexity some sub-system of a particular complexity that sub-system complexity will be repeatedly used by you that is your property which you will use again and again. If somebody else wants to use that he has to pay a royalty to you for that. So, such a thing is called an IP in this industry.

In the design industry IP stands for intellectual property, that is, you have developed a functional unit from basics and that functional unit is available as a code to be integrated into the software of the other person who is developing for his application. It is available as components but we are talking about discrete components in this series of lectures so we will call it a register which is off-the-shelf components available in the market, commercially. Register B is the one which will store the value of the number of 1's. So the maximum number of 1's will be all 1's for n-bits so n so that will be the logarithm of n to base 2 plus 1 and this will be a counter which will be initially cleared and then the clock will go but then clock will not count it up but it will count up only by a count enable thing. The system clock is common to all the registers. Having decided this, architecture of the functional blocks we will go to the state graph of the control unit. It is a very simple state graph. It stays in a state called S 0 a reset state, I can have notes here somewhere…. S 0 reset state, I am go to assume a signal called S, the start signal is going to come. Only when the S comes you start loading the register A and then start counting the number of 1's and then putting the values in register B.

(Refer Slide Time: 8:33)

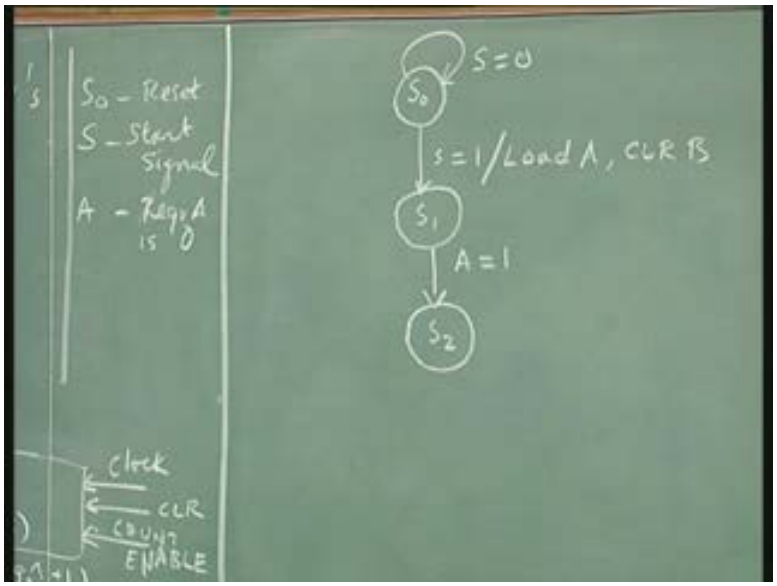


So I am going to assume a signal called start signal. If the start signal is 0 I continue in this state, so S is the start signal. If S is 1 I go to the next state I will call this state S 1 but before doing that when as soon as S is 1 I would like to clear the register B and load the bits into the register A. So, S is equal to 1 will give an output of load A clear B these are the outputs; load A clear B are the outputs when S is equal to 1 in state S 0. In S is equal to 0 when S is 0 it continues to be there, and when S is equal to 1 it goes to the next state giving these two outputs (Refer Slide Time: 9:29). In S is equal to 1, suppose these are all zeros how do you know we have finished the job? I keep shifting the number of bits. I keep shifting the register A and filling the shifted positions is zeros. When all the bits are zeros I have to stop the operation. That means the value of register A is 0 then I end the operation. Or it may be that I am loading all zeros then also I need not do the operation.

So, when the value of the register A is 0 I want to terminate the operation. So I am going to call a signal A. signal A is a signal which will become 1 when the value of the register is 0. So, it is the signal I am defining. All this is I should account for in the architecture, functional units. So A is the signal which gives you the content of register, register A is 0. So, if A is 1 that means I have either completed the operation or there is nothing to count, either way, why I put the same value A is because A is the register and A is the value. Then in that case I go to S 2 state, so I am ready to go to reset and load the next value but S is a start signal it could be a manual signal, somebody pushed a button, if I do not make sure that S is 0 again I may go back and redo the same thing (Refer Slide Time: 11:26).

Supposing I have non-zero value of A and it is going to take just few clock cycles to count the number of 1's still if S is not released from your…… you have not released S the start signal, you make a start signal 1 and you have started the work and you completed the work and S is still 1 I will go back and again it will redo the same thing over and again but I don't want to do that. So, before going back I will have to make sure that the S is being removed completely to 0 then the next time it becomes 1 and again it starts the operation.

(Refer Slide Time: 12:05)



Something like you want to let one person in into the room you open the gate and then you keep it open sufficiently open so lot of people can enter. How do control it? So make sure that you close it and open it again. Every time the door opens only we can leave the person in. you can make an algorithm or give an instruction that just because the door is open you should not come in and only when the door is shut and then it is open you should come in, something like that.

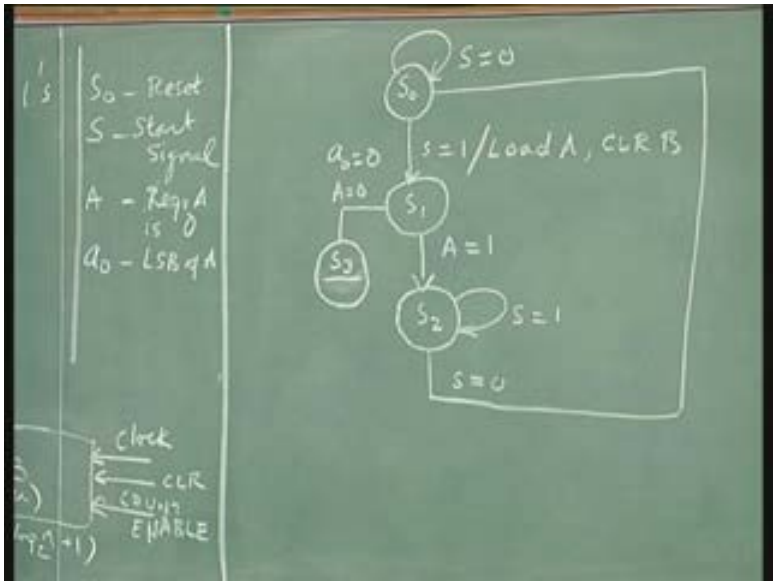
So what I am trying to say is, if I am doing this I is equal to 1 either because the value you have loaded is 0 to start with or you have finished the job, either case I am done I am ready to start the next operation, next count, next loading of a….. loading of next value to register A and counting it again, all of that before I remove doing that I should once remove the start signal and then get the start signal one more time. Every time the start signal comes I should do the operation only once. When there is a start signal, I load, finish count, stop. Again next time the start signal comes I again load the next value otherwise I will be doing the same thing again and again, I will be counting the number of 1's in the same value repeatedly, there is no point in doing it.

So what I do for that is I now see whether S is 1, if S is 1 I will not go out of this loop. That means I have not removed this signal S is equal to 1 because I did not say S is equal to 1 is going to last for one clock period. If I said that it is only for one clock period then I will be safe but I do not know what is the duration of this S, it is a manual push, it can last for several clock cycles and by the time we can finish the operation but still S is equal to 1 so you don't want to go back. If S equal to 0 I will go back (Refer Slide Time: 14:10) so that part is over.

Now the counting part remains. How am I going to count? I am going to look at the LSB. I am going to see if A is equal to 1 that means the value of the register is 0. A is equal to 1 means the value 0. Register A is 0 is called A. So A is 0 means A is not 1 that means A is 0 then I have two choices, I have the bit [loo…..14:50] at the LSB is 0 or 1. If it is 0 I do not count but I shift, if it is 1 I shift and count so I two possibilities. So A is equal to 0 is a non-zero value of the register A and my last bit I am going to call it a 0 is equal to 0 the last bit 0 LSB. a 0 is the LSB of the

register A. Non-zero value and LSB 0 I don't count I shift so I go to the state S 3 and in S 3 I issue a command of shift A, I shift A.

(Refer Slide Time: 15:52)

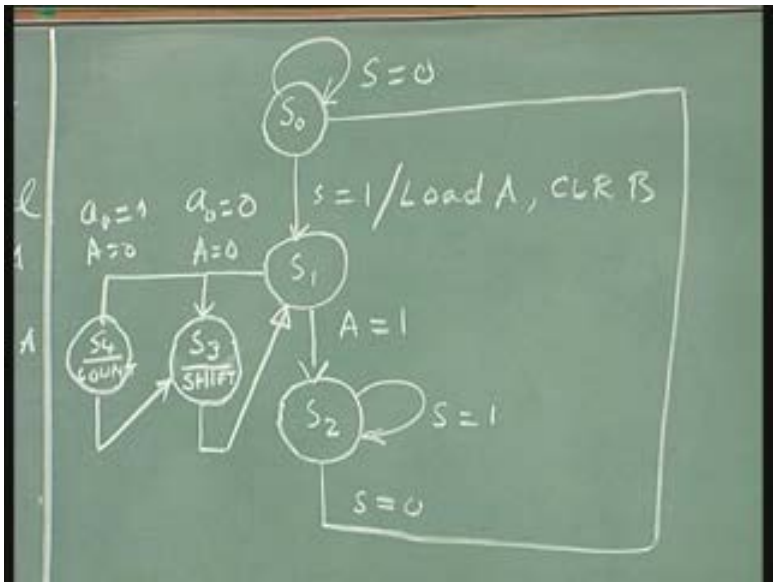


So this signal is the shift signal (Refer Slide Time: 15:58) then go back see if this A is equal to 1. On the other hand, if A is 0 and a 0 is also 1, I should count 1, [b…..16:24] has to be incremented by 1 then I shift and then I go back so I have to……. A is equal to 0 a 0 is 1, LSB is 1, I will go to another state called S 4 and here I will say count and then I go here and then I go here.

I have not drawn it very neatly I know but I have developed, you can redraw it and then make it look like what you see in books. That is not the point; the point is to know how to do it properly. Now let me go over the whole algorithm one more time quickly. We are waiting for a signal call start and when it is not there we keep in that reset state, we keep the system in the reset state and we find that S is equal to 1 suddenly and then before we go to the state S 1 where we start the operation, before doing so we load the register A with the value for which you want to count the number of 1's and you have to clear B because you have to start with zero count and then you find the value of A. If it is 0 there is nothing to count, we have to stop. But before doing so I can directly go back and wait for the next start but I don't know whether the earlier start has been removed. So in order to do that I am introducing an extra state called S 2 in which I will wait if S continues to be 1 and when S becomes 0 I return.

After loading A if it is a non-zero value I have to look at a 0. If a 0 is 0 non-zero value, if a 0 is 0 I don't do anything but simply shift it to one position right. So non-zero value and LSB 0 I shift it and then again look at the value, if it is 0 I continue otherwise I go back, this way (Refer Slide Time: 18:35). On the other hand, if the non-zero value and the first bit is 1, a 0 is 1, LSB is 1 then I do this operation, shifting has to be done, verification has to be done but before all that I have to count up by 1.

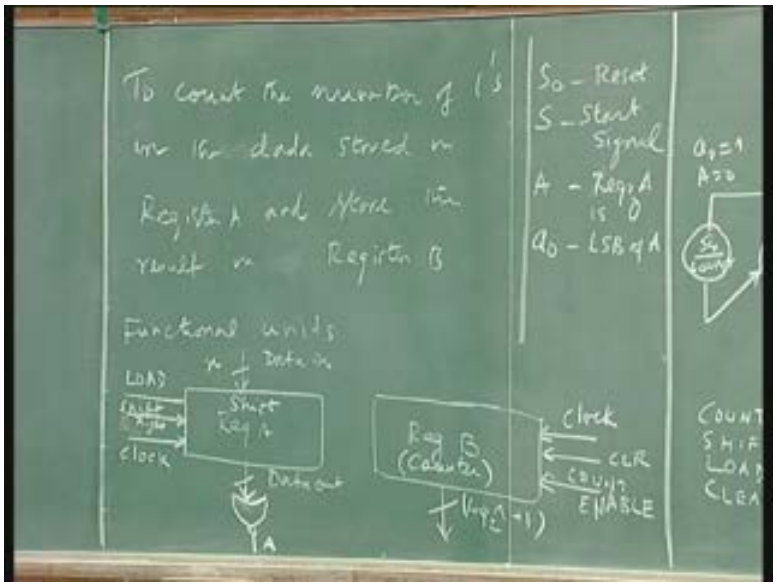
(Refer Slide Time: 18:49)



Any questions are here? Is it a Moore machine or a Mealy machine? It is a combination. There is nothing like Moore machine or Mealy machine because these two are Moore outputs and this is the Mealy output. The system need not be Moore or Mealy. So what are the outputs now? Count, we have already defined this 'count_enable B shift_right A load_A clear_B. Two other inputs we have not taken into account, we require here. I am monitoring A is equal to 0, the value of the register, it has to come from the functional unit that much….. Remember we have to go back and forth between architecture and the functional units. As I draw the state graph I will find out the limitations of my functional units inadequacy of my functional units and come back.

Now what I have not done is, how you are going to verify whether A is equal to 0 or 1, I have not thought of it till now. I did not know it was required when I drawing this but now I am drawing this circuit I know, when I draw this state graph I know there is a need for it. So how am I going to do that, it is very easy because I have the output data, I put a gate here to find out if it is a 0 or 1 so I will have to now give this out of this (Refer Slide Time: 20:39) so I will have……. if all the bits are 1 how do you know if it is a 1 or 0, what gate will give you? NOR gate can give you; anything can give you actually depending on how you connect it. Supposing I give a NOR gate if everything is 0 it will be 1. So, that's what we want; A is equal to 1 all are 0.

(Refer Slide Time: 21:14)

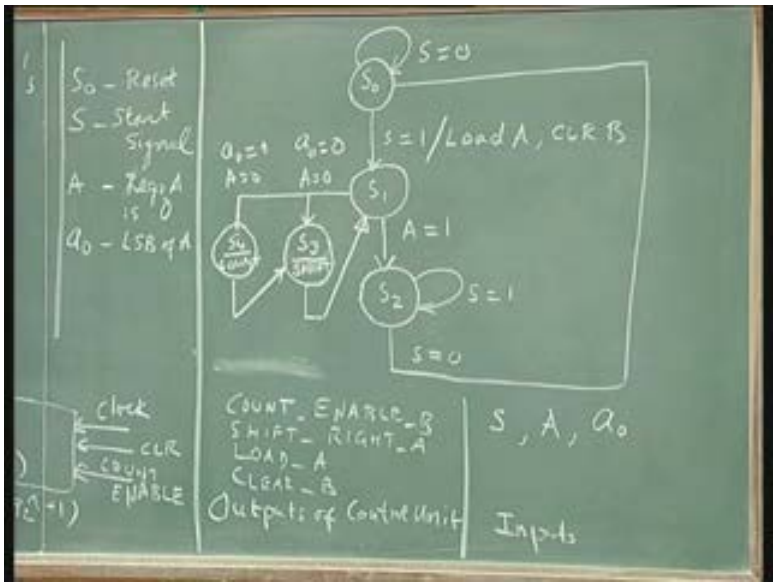


This extra functional unit requirement was not evident at the time of drawing the architecture part or the functional unit part. Only I went to the state graph I remembered or…. not remembered but I found out that something like that is required and at least the way in which I am doing this problem. Of course you can do it some other way more efficiently, fine.

Other thing is, I want to know a 0 that is easy because I have all these bits here, all the eight bits are available here, eight bits and n-bits are available to look as the LSB. So LSB is only here so a 0 is here, so both of these things come from here. So the two extra inputs which we did not think of in this case the LSB of the word and the 0 value, 0 value is called A and A is equal to 1. That means register A's value is 0, a 0 is the LSB because all the bits are available so the least significant bit you take and put it.

These are the outputs of the control units; inputs to the control unit are S, it starts with which is the input of the control unit, value of A and a 0. So now rest of the problem is simple. I have a state graph, inputs and outputs are clearly defined, present state and next state, I draw a ROM information, PLA implementation, mux implementation, and whatever implementation you can imagine of. So the most important problem is to understand it, partition it to the extent possible of what is available already off-the-shelf or the previously designed functional unit with you or a sub-system with you, identify the input output signals of this unit and see whether it can be used in the controller, start drawing the state graph to control the functional units in the sequence required to realize the specifications, go back and fourth couple of times, iterate between these two to find out all the signals are accounted for, all the signals are required as control unit inputs or accounted for in the architecture and all the functional outputs coming out of the controller can be accommodated into the functional units so that they can realize what is required, they can implemented what is required then when you are done and you are satisfied with the…… and the algorithms also should be right, don't worry only with the signals and finally we will not be doing what we required to do then it is no good.

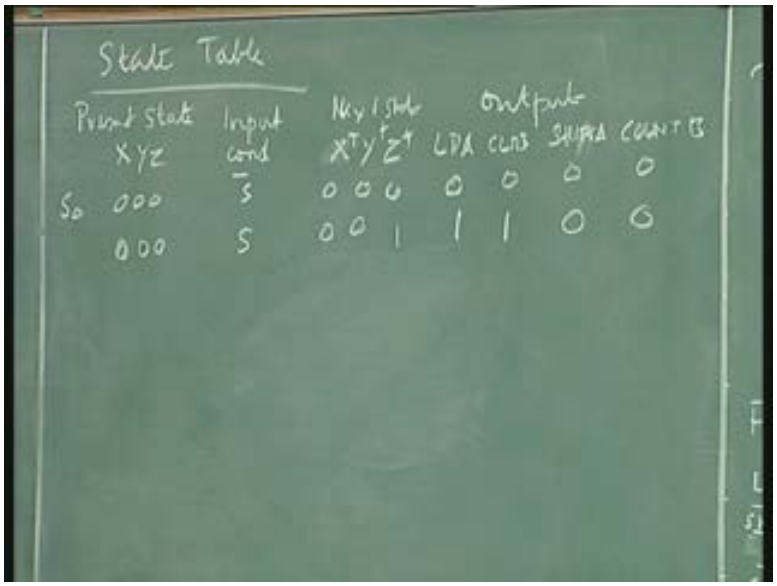
(Refer Slide Time: 24:50)



Then you have the state graph. I am not going to implement it, you have done it hundreds of times at least if I assume that you are doing some of these exercises at home. What I am going to do is simply write the state table and leave you to do the rest of the work. State table would be; present state, we will have the standard nomenclature S 0 is 0 0 0, S 1 is 0 0 1, S 2 is 0 1 0, S 3 is 0 1 1 S 4 is 1 0 0 and let us call this x y z, here I am using a b c too many times.

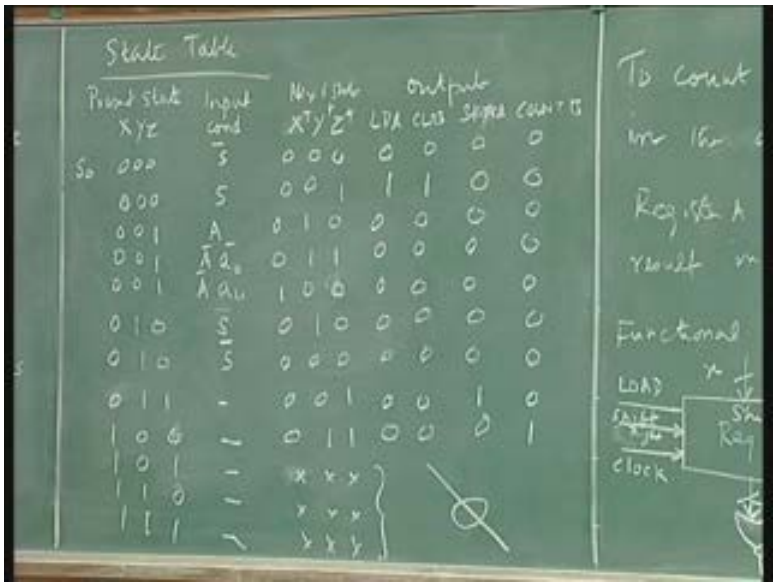
So present state x y z input conditions, next state x plus y plus z plus outputs. outputs are load A Clear B Shift A Count B so present state S 0 is 0 0 0, S is equal to the condition S bar remains here 0 0 0 and S here is 0 0 1…… and here no outputs, (Refer Slide Time: 26:36) and here the output is 1 1. Present state is 0 0 0 if start signal is not there it will remain in the same state without any outputs, if S is equal to 1 go to the next state 0 0 1 state and give the outputs load A and clear B.

(Refer Slide Time: 26:58)



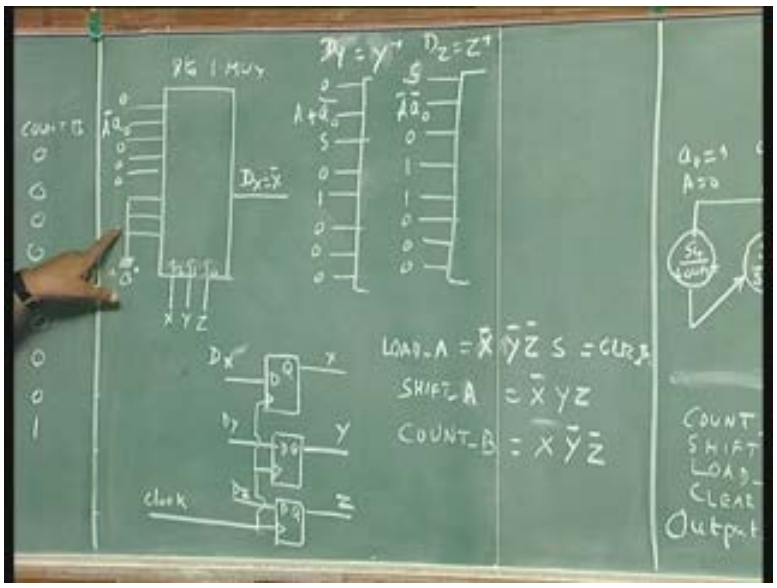
Then S 1 0 0 1 three possibilities are there, I will just simply write it, A A bar…… 0 1 1 1 0 0 no outputs, no outputs in any of these, so please tell me if I made a mistake in drawing this when I write this table fast. Then 0 1 0 S bar again S remains is no outputs 0 1 0 goes to 0 0 0 again no outputs. 0 1 1 is without any thing….. 0? Which one? Probably this is a S bar? S bar goes to reset and S remains here. 0 1 1 without any condition (ms) it goes to 0 0 1, 1 0 1, 1 0 0 without any condition it goes to 0 1 1……..(Refer Slide Time: 29:14) 1 0 1 1 1 0 1 1 1 not used. You can put don't cares and all the outputs make 0.

(Refer Slide Time: 29:31)



So you can implement it using mux as an example. Use muxes and three flip-flops, d flip-flops and mux solution. The implementation of this mux based solution for this design is shown here, we are using three flip-flops d flip-flops with q and q is being marked as x y z, these (Refer Slide Time: 30:07) are the state variables; D x D y D z are the inputs to the flip-flops all controlled by the clock. now each of these D's comes from a 8 to 1 mux, for the D x x y z are the selected variables o the state present state and the next state information has to be marked here we know how to do that, we get these values; for example, first input is 0 then it is A a bar a 0 0 0 0 0 0 0 etc. Similarly, for D y these are the inputs to the multiplexer 0 a or a 0 bar S 0 1 0 0 0, the third mux D z the inputs to D z are given by S a bar a 0 bar 0 1 1 0 0 0.

(Refer Slide Time: 30:59)



Outputs are very simple there are only a very few outputs, load and clear are the same which are X bar Y bar Z bar S, shift_A is X bar Y Z, count_B is X Y bar Z bar. So we have a three input solution, three flip-flops each driven by the multiplexer whose inputs are connected like this and outputs can be got by simple gates and we have the entire implementation. With this example we have seen the system design, we talked about top down design. The approach is to start with a problem, separate the functional unit which is called architecture, from the control function get the functional unit implemented as far as possible from either the components available to you or the IP course intellectual property course developed by you earlier.

You have to be very careful identifying the input output requirements of these functional units then go to the state graph of the control unit. Let us go through the algorithms systematically and draw the state graph. As we proceed to the state graph we will find that some signals are not defined properly in the functional units, we might have to go back and fine tune the architecture and a couple of iteraces back and forth, then we will have the final state graph. So the functional units are already available, the signals, we should know what are the control signals available, of course the data input and data output are known, in addition what are the control signals available and how you are going to use that. As far as the control unit is concerned it turns out to be a state graph and we know how to implement state graph as seen in the last example.

With this lecture we conclude this course on basic digital system introduction, introductory course on digital system. We have covered large amount of material in this course. We talked about the digital signals, what is the difference between continuous signal, discrete time signal and digital signal, advantages of processing signals in the digital domain, how at the same time we cannot do away with analog signals. So the real world is analog both the input and output so input signals analog signals……, we do minimum amount of signal processing analog domain, convert them into a digital signal by a process called analog to digital converter and then go through the digital domain, it can be simple circuit the gate, simple combinational logic, a simple sequence logic or some sort of a combinational sequential logic so some combination we have seen or some of the systems we talked about or it could be much more than that. Finally the output will be digital signals, and digital signals, if they have to drive the analog world but again convert them back to analog signals by operation called digital to analog converter.

So we have analog world, analog digital converter, digital processing, digital output, digital into analog converter, convert the signal back to analog into the real world. We have seen the basic building blocks started with the different types of gates, primary gates, secondary gates, universal gates. We talked about logic functions, how to build simple logic functions using gates. We talked about the truth tables, min terms, max terms, sum of product and product of sums. We talked about Boolean algebra for simplification of the logic functions so that we can save on the number of gates used and we talked about Karnaugh Maps.

Then we designed some simple combinational circuits functionally like adders, code converters may be, parity checkers or parity generators, then we defined the sequential circuit, different sequential and combinational circuits. First sequence is where the circuit in which there is a storage element, memory, is it required in any real time system any real life system. Then we discussed the basic building block of latches leading to flip-flops, talked about different types of flip-flops, different types of triggering mechanism, master slave and edge triggers.

Using the flip-flops as building blocks we talked about counters, registers, shift registers and simple sequential circuits. When we say sequential circuits it is not necessarily only the sequential building blocks like flip-flops and counters, we can also have a driving combinational logic, the total system is sequential. That means we have a sequential system which goes from sequence, we have a sequence of states but when it goes through that it will be driven by a combinational logic. The steering logic is combinational so the integrated circuit the total circuit is a combinational sequential combinational. And then we talked about, instead of going through the gate solutions a simple building blocks solution, we can go to higher levels of integration.

We talked about integrated circuit evolution, from small scale integrated circuits to medium scale, the large scale to a very large scale integrated circuits. We talked about MSI and LSI components from combinational and sequential logic. We talked about multiplexers, decoders. We talked about PROMs Programmable Logic Devices, Programmable Logic Arrays, Programmable Array Logic and also how to use these in digital design, use that for combinational design or use in the steering logic of sequential circuits. We also said we can also do a similar job with the sequential circuit. Instead of using flop-flops we can use registers and counters to design sequential circuits and steer them using combinational building blocks.

Ultimately we can have a single IC a Programmable Logic Device which has both combinational and sequential components in it to implement simple digital systems. Then we took examples of simple code pattern generators I mean how to identify the pattern, identification of a pattern from a series of bits, sequence detectors as they are called. Then we talked about other systems, slightly more involved systems like traffic light controller. We reduced the problem to a state graph and how to do a state graph implementation using different types of approaches, multiplexer based, PAL based, PLA based and also using registers and counters instead of only flip-flops for sequential building blocks.

Then we got to the concept of top down design. Because in the real world somebody gives you a system specification he doesn't bother, he doesn't tell you how to do it, we have to do it the most efficient way. So, best thing is to partition the problem. Look at the problem as functional units and control functions. The functional units are already done and there is no point inn repeating it, no point in designing it all over again, no point in reinventing the wheel. Choose these functional blocks, identify the input and output requirements and availability and do the control function alone.

So it is a question of drawing the state graph properly of the control unit. It is a question of identifying the functional blocks along with the input output signals, drawing a sequential circuit control unit, state graph of a control unit, taking into account all the input signals and output signals which will control the sequential building blocks the functional blocks and then we have to integrate between these two by means of signals called command signals and status signals. And once we know that the main job of design is the understanding of the problem in complete in its entirety, completely understand the problem, the exact specifications as required by the user, the client, as a designer I should be able to given an unambiguous solution.

So, interpreting the specifications properly and drawing the state graph is the most critical part of a digital design and in the case of an integrated system and in addition to that to identify the

proper functional units the most efficient way. Having done that we have the rest of the tools straightforward, we know how to implement a Boolean function in gate form, in MUX form, in PAL form, you know how to draw a state graph, you know how to implement the state graph using different types of logic so all these are standard procedures. Of course you need to do that in order to reliably build the circuit, test it and make it work. These are non-trivial operations but still the most important aspect of the design is the identification of the problem, identifying the specifications completely, understanding the specifications completely and identifying the input signals and output signals, identifying the suitable functional blocks if they are available or design them using the functional building blocks building basic building blocks and then identifying and coming up with a state graph and the steering logic and implement the state graph using steering logic.

We have not seen some of those things like how to…. We have used arbitrary assignments of states always, sequential binary sequential. Sometimes there are some guidelines we followed in order to reduce the number of gates required, hardware required. Likewise we have not done a state reduction, we have given a state graph, you evolved a state graph and so on. What if there are more states than required? There are procedures. But this is all trivial things, we can always learn these things.

So, in this course we have seen an entirety of, starting from the basics of almost from the definition what is a digital system, what are the various functional units, what are the building blocks, what do you mean by combinational logic, what is meant by sequential logic, what is mean by systems all that, we have seen that the entire…… up to designing of simple systems. Of course I don't, when I say simple systems it is only because the limitation of time and limitation of the amount of material that can be transferred in a classroom. If more complex systems are more extensive I won't even say complex, more extensive system is not all that difficult, all you have to do is to do more of it which means it takes more time and more sort of more organized work which cannot be done in a 1 hour lecture in a classroom. But we can easily pick up, the concepts are not going to be very difficult, the concepts are same.

We have learnt all the concepts required and of course if we go to very large scale integrated circuits like computers and microprocessor design and digital signal processing design and all these very sophisticated digital systems we may have to go for other tools namely the analysis tools, design tools and synthesis tools. But otherwise however complex a system may be for a given application [we are……. 43:56 all] to design it and to implement it using the type of components you are asked to design with. So what is left? Lot more of course. There is nothing like you have learnt all. Basics are clear and we have the capability to design simple circuits. when I say simple again it is not simple in the sense of there are more advanced concepts which have not been told to you, it is not in that sense I am saying it is simple.

Circuits are small because we are handling them in a reasonable level of complexity in a classroom. More extensive circuits we should try for practice. Certain aspects we have not covered as I said the reduction of the number of states, assignment of states, assigning a binary value for states so that using some guidelines, reduce the number of gates or the hardware; these are some of the things you can learn by yourself. But one important aspect of a digital system you have not covered is what is known as asynchronous sequential circuits. A circuit can be a

combinational circuit or a sequential circuit. A sequential circuit goes through states various states and the behavior of the present state depends on the present state input and the past output. Now, if it is all done with reference to a clock cycle we have a clock period, the clock has a basic driving force, with that clock if you are talking about then it becomes a synchronizing circuit. If something happens in between clock it will not be recognized until the next clock pulse occurs. Mostly it is edge triggered. From this edge to next edge something has to wait. Supposing if we are interested in getting things over as fast as possible, a clock is supplied there is no clock of course in this case, there is a circuit here and if something happens and once that is completed you want to take the next step but it is not a combinational circuit it is a sequential circuit but don't have to be controlled by a clock, such a circuit is called asynchronous sequential circuit.

Asynchronous sequential circuit has an advantage that, we don't have to wait for a clock period to be over before we can take the next step. Once the step is to be completed the next step should to be taken but that requires a little more analysis, little more detailed analysis and design, just a slightly more complex in terms of concepts also. So that is one thing we have not learnt. Otherwise we are now fairly comfortable with the type of designs which we normally encounter and digital world and most of the circuits by the way are synchronous sequential circuits. Asynchronous sequential circuits are very few and are used only in extremely rare cases when the speed is of essence or there are other reasons for going for asynchronous sequential circuits.

[b…..46:30] that we are having a basic tools required for understanding, analyzing and designing digital circuits, digital sub-system and simple digital systems, we can very easily extend these concepts to design bigger and larger systems without having to learn any extra concepts; more practice, more examples, more training.

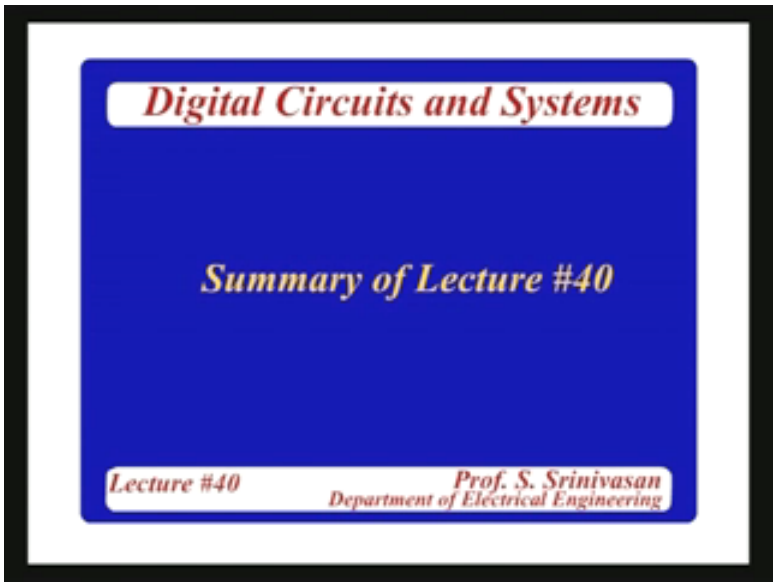
So finally to summarize this course; we are in the digital world today. Everything is digital around you. Your television is digital, your control is digital, traffic light controller is digital, washing machine is a digital controller, your remote control is a digital control. So we are talking of digital today in this world and we are in this real world, analog cannot be dispensed with but we reduce the analog signal processing to the minimum possible extent.

So what are the issues involved here. the issues are, how bigger and biggest circuits, larger and largest circuits with the minimum amount of cost, that means we will have to put lot of functions in a simple circuit, cost reduction, size reduction, power reduction, performance improvements (peak), highest performance possible in terms of speed and other qualities of performance, the lowest cost, the smallest area and that is the world to which we are moving and you have taken a first step, it's a very very firm first step which is not very trivial or a shaky step. So if you have really gone through this material completely and understood all these lectures, the concepts and worked out the unfinished examples in the class you are ready for the design to get into the real world to digital system design.

Many exercises that were given were left out in the middle and I asked you to complete them. other then that if there are more exercises all you have to do is to take a standard book, of course I have given a reference book in the beginning, there are others in which……, digital systems in so many books today, take any standard text book on digital system design and then work out the problems, back to the book problems. These are the exercises; there is no need for me to give

separate exercises because there are at least half a dozen good books on basic digital design today, you can take any one of them and then start working out these problems back in to the book then we will completely be confident of the whole concept, be ready to face the challenges of the real world where one has to design complex digital systems. Thank you for listening to this course and I hope you have benefited by this course.

(Refer Slide Time: 00:49:26)



(Refer Slide Time: 00:49:44)

(Refer Slide Time: 00:49:51)