Digital VLSI System Design

Prof. S. Srinivasan

Department of Electrical Engineering

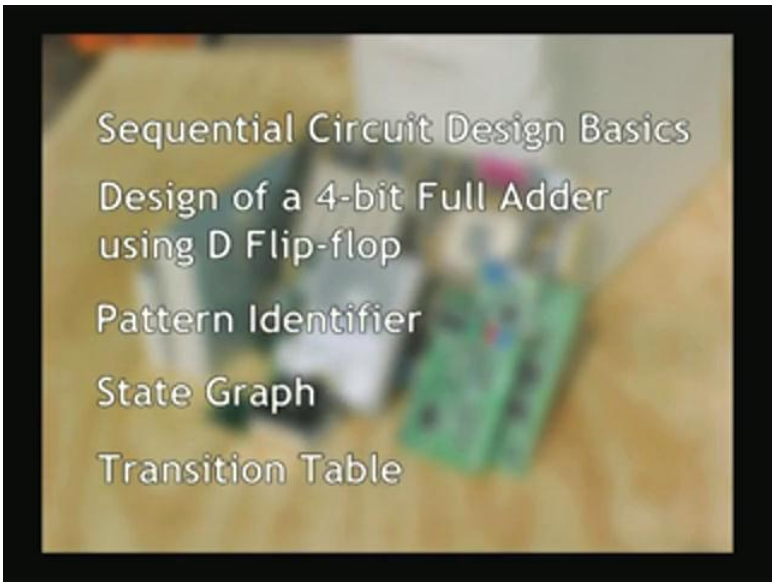
Indian Institute of Technology, Madras

Lecture – 8

MSI Implementation of Sequential Circuits

Slide – Summary of contents covered in previous lecture.

(Refer Slide Time: 01:10)

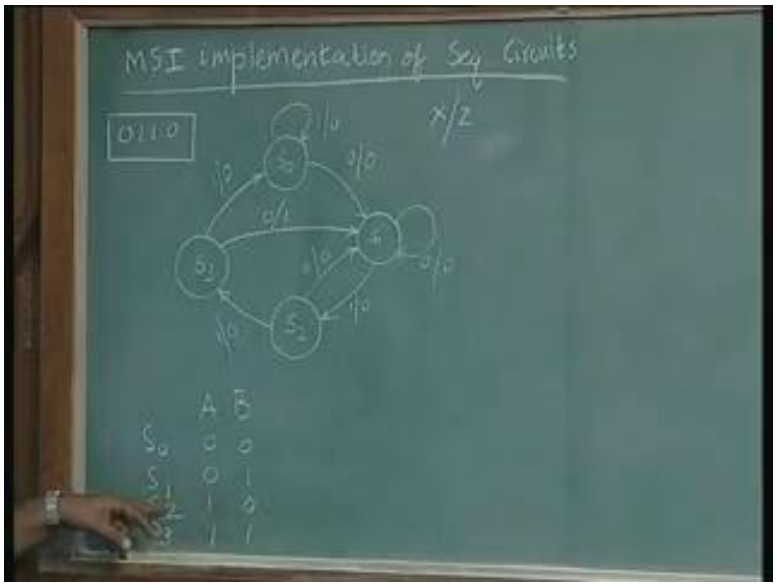


Slide – Summary of contents covered in this lecture.

(Refer Slide Time: 01:38)

In the last lecture, we developed a methodology to design a sequential circuit. We took a simple example of a sequential circuit meant to detect a pattern 0 1 1 0 from a serial bit stream. For this circuit we developed the state graph.

(Refer Slide Time: 01:58)

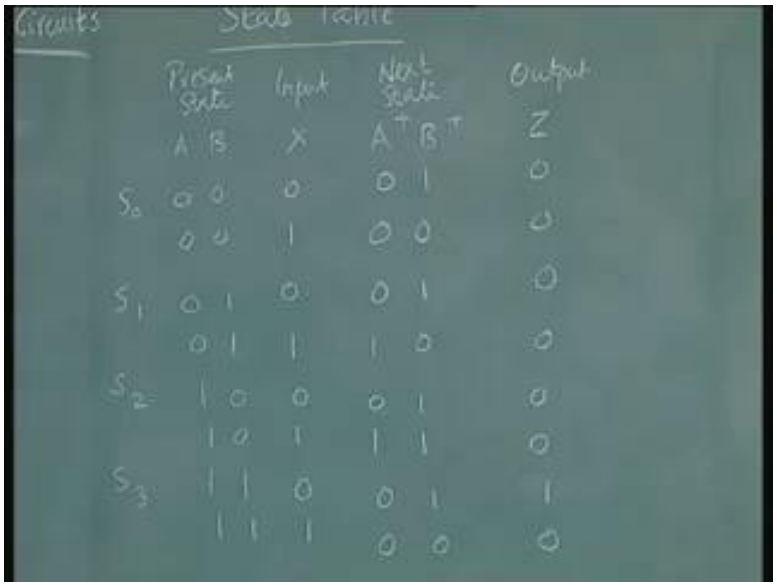


This was the state graph we got (Refer Slide Time: 02:22). There are four states in this; I assigned state variables such as A and B. Further assign these values to this A and B as given here: s_0 0 0 s_1 0 1 s_2 1 0 s_3 1 1. There are only four states. The input bit stream we indicated by X

and the output z. Whenever a pattern 0110 is obtained, z the output will be the one corresponding to the last bit of the pattern and all other time z will be 0. We also set this as an overlapping pattern. That means, if another 1 1 0 occurs, this 0 will be taken as the first bit of the next pattern also. Anyway, this is a simple review because this sequential circuit design would have already been exposed to you in the previous course on Digital Systems. In order to keep the flow and to define some underlying concepts we took the simple example. Since, the focus of this lecture series is more on implementation using ICs and final implementation of IC systems and VLSI systems; we would not like to take a look at the gate level implementation of the circuit. We did that in the last lecture. We took a couple of Flip-flops: the Flip-flops A and B was driven by the circuits involving gates. We went through the karnaugh map reduction and all that normal procedure. As I said this is more of a review; when you design for an IC, we will use more and more complex hardware or let us say hardware functionalities to replace gates, so that the design becomes simpler.

Today, we will see some of the mapping techniques of implementing the circuit using MSI components. The combinational logic will replace the gates of the combinational logic with components such as multiplexers and ROMs and PLA. That will be the focus of this lecture. So let us get the state table which we have already got in the last lecture, rewrite it now.

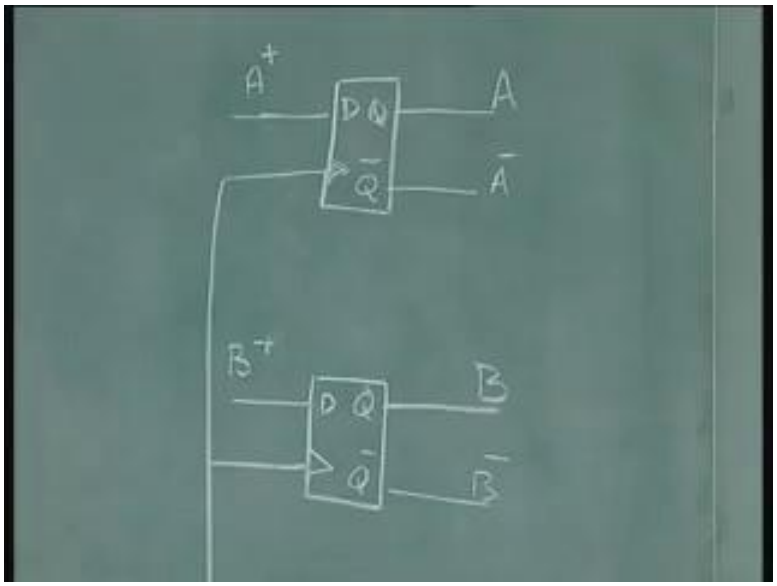
(Refer Slide Time: 04:58)



In this state table, the present state variables are A and B and as I said the input is X. In the next state variable, the output z and there are four present states: s_0 0 0, then s_1 0 1, s_2 1 0 and s_3 1 1. Hence, from this state table and (Refer Slide Time: 05:55) from the previously discussed state graph we can write the following state table. If X is 0 what happens in next state and what is the output? X is 1, what happens? Similarly find for s_1, s_2 and s_3. Without spending a lot of time since we have already done this exercise, I can write this as 01 0, 00 0, 01 0, 10 0, 01 0, 10 0, 11 0, 0 1 1 and 0 0 0; this is 1, output that is a_1 (Refer Slide Time: 07:08). This is the next state table and the output information corresponds to the present state input. This is the same table which you got the last time or you could also simply get it from here. For example, if you are in state, s_0 input is 0, output which is also 0 and this state goes to s_1 which is 01. So 0 0 0 01 output 0, like that one can do all of the 8 rows. Instead of going through the reduction using karnaugh maps as we did in the last lecture, using gate implementation, I am going to use multiplexers as combinational building blocks. Since you saw the implementation of combinational logic using multiplexers, we should able to easily fit in that. We know that any combinational logic with the definition of the input output relationship can be implemented using an appropriate number of multiplexers.

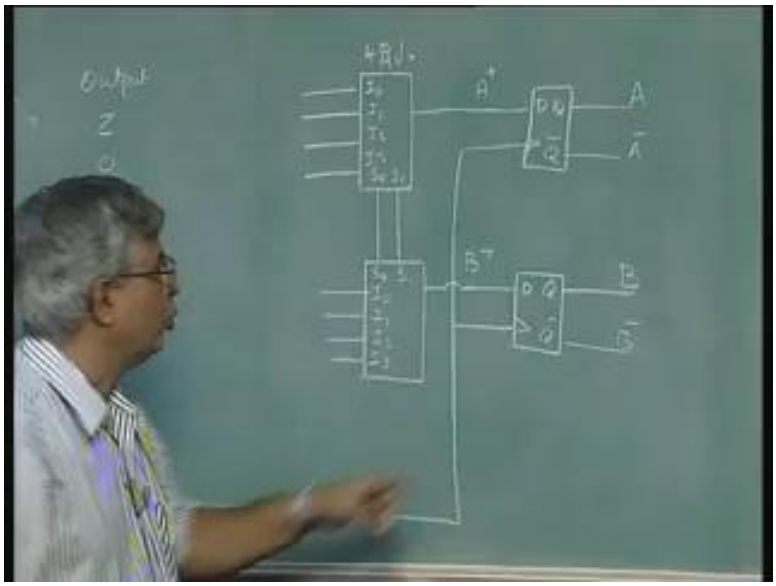
Let us try to do it here. It is a simple procedure here, since there are 2 stages there are 2 Flip-flops, that fact is not going to change whether it is a gate oriented design, or a MUX oriented design, or a ROM oriented design or a PAL oriented design. This number is going to change the fact that four stages required two Flip-flops. Let us call this Flip-flop A and B as we have done here in this case.

(Refer Slide Time: 08:20)



This is Flip-flop A, output is A. Flip-flop B, output B (Refer Slide Time: 08:35). Q is the A and Q bar is A bar. Similarly, Q is for Flip-flop B and Q bar is B bar. D of the Flip-flop we will call it DA or A plus, the next state of information will be, for the D Flip-flop the output follows the input; that means, if you give the next state information, here we take as the present state when the clock arrives and that is what is called A plus, if you want call it DA that is also fine. Similarly, call this D and B plus, driven by the common clock. This is required for the four states, we need this type of 2 Flip-flops working. Now the question is, if the four values of A and B 0 0 0 1 1 0 1 1 will occur based on the present state and the input given, that information has to be given to this input of these Flip-flops D A and D B, only then they will change state accordingly. Who is going to give the input? In the previous design the combinational gates were responsible for giving that input.

(Refer Slide Time: 10:00)



In this case, I am going to make multiplexers give that input. I am going to do is to put a multiplexer here; another multiplexer here and look at what is the size of the multiplexer (Refer Slide Time: 10:20). The size of the multiplexer is decided by the number of different outputs and different inputs. There are now going to be 2 state variables and that means, there are going to be 4 combinations of inputs. For each combination of input we should decide the next state. If you have 2, 4 to 1 multiplexer here, with two control lines (Refer Slide Time: 10:45), for each of the 4 states 0 0 0 1 0 1 1 we can find out the next state, so the next state information is A plus, for each of the present state information will be given from 4 to 1 multiplexer.

The size of the multiplexer is decided by the total number of states in the circuit valid design. In these cases, the total number of states in the circuit is 4, so I need 4 to 1 multiplexer. If I had a total number up to 8, I should have 8 to 1 multiplexer and so on. Likewise for the second output DB or B plus if you want to call the B plus or DB, it does not matter, the present state information s_0 and s_1 will give the corresponding next state based on the input condition. Again I will give four choices of s_0 and s_1 for four different states and each of the states will correspond to an input condition which will be the next state. Now that means I have the I_0 I_1 I_2 I_3 and s_0 s_1 are the present state variable. The present state variable with a given input condition gives you next state outputs for these two multiplexers. How do I arrive at this present state information?

The present state information is available with these 2 Flip-flops so I need to connect this s_0 to A and this s_1 to B. s_0 is nothing but the state variable A and s_1 is nothing but the state variable B. So now look at the table (Refer Slide Time: 13:05) here it is A B 0 0 0 1 1 0 1 1. For each of these conditions I will take the appropriate input of the multiplexer and see whether the input is connected to X is equal to 0 or X is equal to 1 based on the next state will be decided because the next state will be decided upon the present state and in the present state based on input condition X. For example if you take s_1 as the present state 0 1, I will not able to decide the next state unless I know what X is. What the X is will have to be, this information have to be given and that will be given through these inputs for that corresponding state. Now it is a very simple procedure, you look at the input condition 0 0 if the present state is 0 0 what is the next state? The next state is for Flip-flop A first, multiplexer A corresponds to Flip-flop A multiplexer B corresponds to Flip-flop B (Refer Slide Time: 14:11). Let us look at the multiplexer A to start with. For a present state 0 0, the next state A plus is decided by the input where the X is 0 and hence the output is 0.

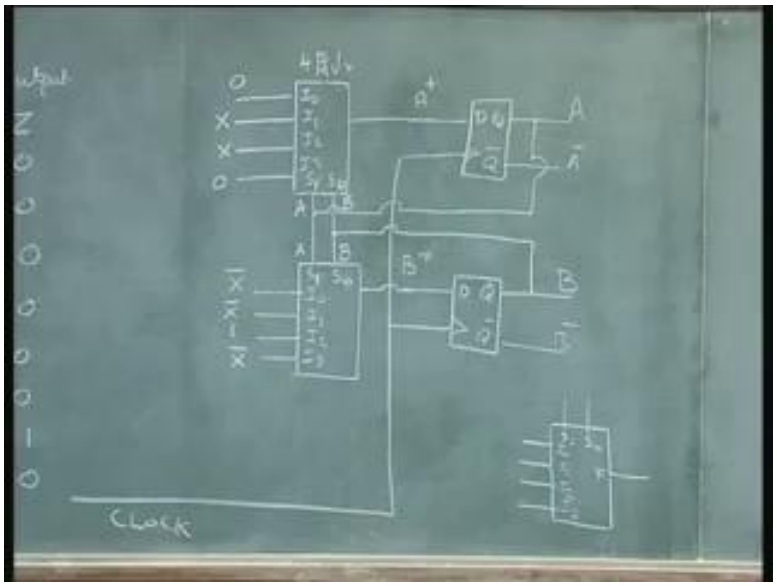
The next state 0 X is 1 also the next state 0; that means; irrespective of the input the next state of the Flip-flop is 0 so I put 0 here (Refer Slide Time: 14:30). You go to the next, the present state is 01, s_1 corresponds to the 01 combination which corresponds to this input I_1. In this case the present state is 01 and the next state of the Flip-flop A depends on the present state. The present state is 0, with the present state input 01; if the input is 0, the next state of the Flip-flop A is 0. If the present state 01, the X input is 1 then the next state conditional Flip-flop is 1. That means, in the present state 01, the next state A plus is decided by the present state X. If X is 0 this 0, if X is 1 this is 1. That means connect X to this (Refer Slide Time: 15:18). X is 0 this will become 0, X is 1 this will become 1. The previous case whether X is 0 or 1, this should be 0 I put a 0 here. Moving on, for the present state 10, which is s_2 condition I_2 input. The next state of Flip-flop A reserve by again X here, if the X is 0 the next state of the Flip-flop A is 0. If X is 1the next state of the Flip-flop A is also 1 again it is same as X, it follows the X. In the last case s_3 11, irrespective to the input condition is 01, the next state of the Flip-flop is 0; I will put a 0 here (Refer Slide Time: 16:05).

This is how you design the combinational logic which will drive the Flip-flop A. Now without having so much of explanation which is already done for one case, I can write simply for the second Flip-flop is X bar, X bar will make it. X is 0 the next state is 1, X is 1 next state is 0 so it is X bar input same here. (Refer Slide Time: 16:37) The third case, present state s_2 the next state

is 1 irrespective of X is 0 or 1. So it is 1 and again X bar. This X is a very simple input now I can replace this by X which is input is available to you, X bar can be got from X by using an inverter I do not have to show it in this drawing.

That is only a convention. Yes if you want to call it s_1 s_0, it is fine with me. I am sorry I should have really not used s_1 s_0 because, this s is a state variable. State s_0 state s_1, state s_2, and state s_3. These are selectors since S is same for state and selector, there is some confusion here, but this is not a big issue. Here it is a convention; it is the highest MSB LSB convention. It does not affect the design in any way. The connection is not changed and all your saying is that I should put s_1 s_0 not s_0 s_1? Actually I should not use s at all because it is confusing. S is used for selectors as well as for states; maybe I should have used some m_0 m_1, something like that is fine. As long as you understand these are the selector inputs and there is no need for, by a multiplexer there will be like this.

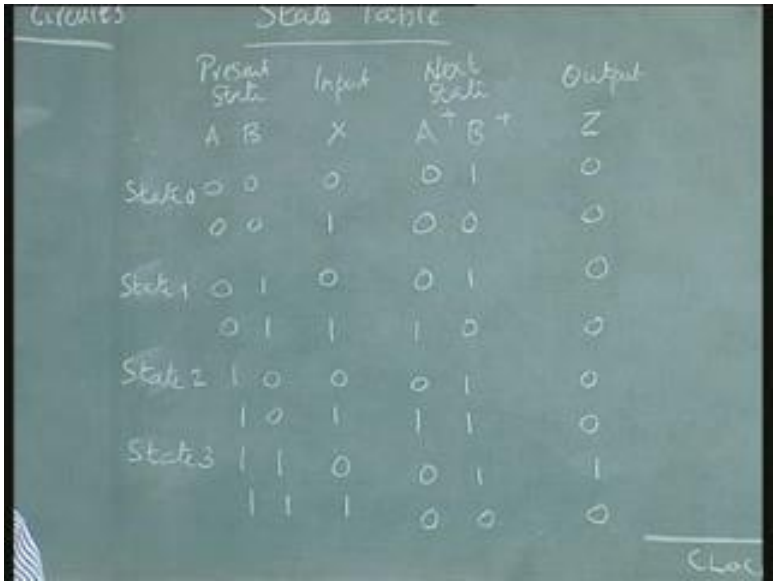
(Refer Slide Time: 18:45)



They will not tell you anything, of course there will be power supply. If you leave the power supply out there will be so many pins. These are the input pins, these are selector pins and these are output pins. We designate this as $s_1$and s_0; I_0 I_1 I_2 and F; designations we give for these pins. That is why this s_1 s_0 came here because these are selector pins, these are input pins, this is an output pin and because of that we got this s_1 s_0 interchange. This is not to be confused with the s_0 s_1 s_2 s_3 because of these s we are not using, it is in the equation. This is only to indicate the

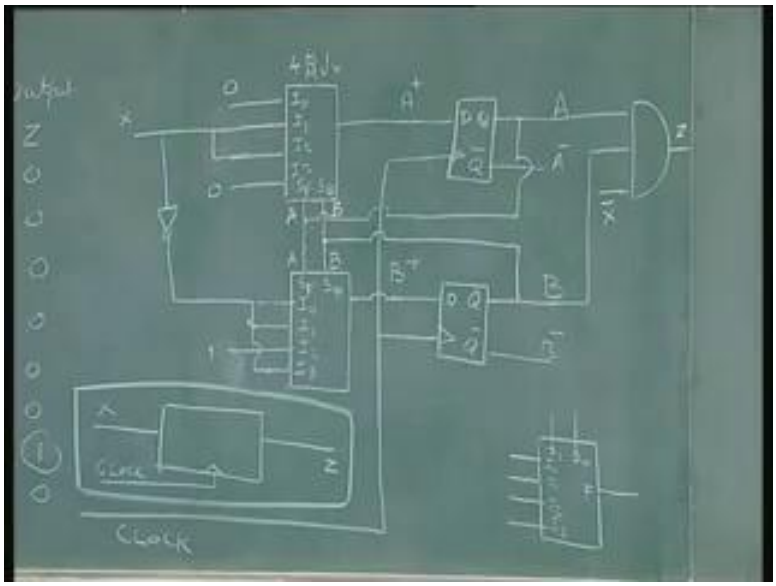
condition so if it down like this s_0 you have to say state 0 (Refer Slide Time: 19:30) so that you can completely avoid that confusion. There is no conceptual or design variation here, it is only a convention, is that alright?

(Refer Slide Time 19:50)



We are not making any design error, are we making a design error? Let us not discuss the convention. Of course convention is there in the book; I know and understand that is why I explained all this very clearly now. The convention is s_1 s_0 that is why you pointed it out. Well taken, I changed it, but we will drop it because I do not want to confuse the viewers of this lecture that it is something to do with design; as it is nothing do to with design. The design is correct, my connections are right. The conventions also had to be respected and I respect the conventions, as well as your sentiments, thank you.

(Refer Slide Time: 20:35)

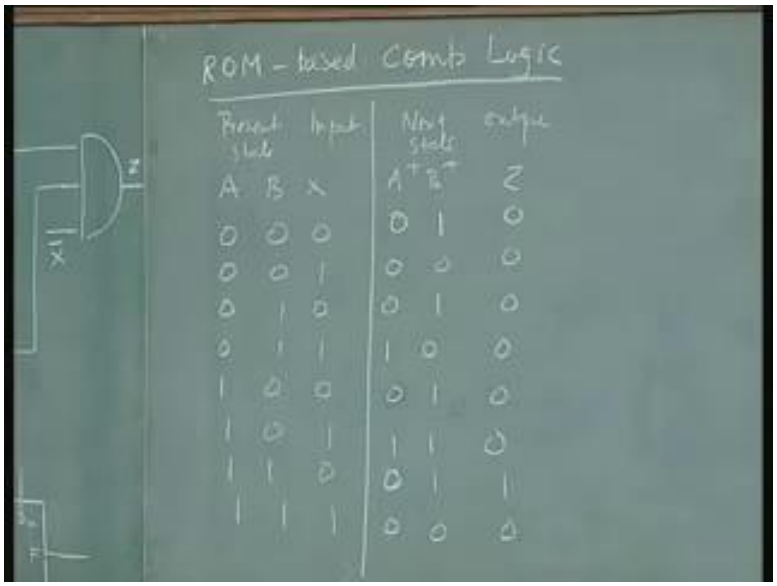


There is only 1 output of 1; all others are 0. That output comes when the input is 11 with an input 0. This is output z can be written as A B X bar. Since X bar is available here you put A B; this output z and X bar is here (Refer Slide Time: 21:19). All I am saying is you take this X as input to a system connected here and the same thing will become X bar and this is 1. This is the design and there is only one input one output because, after all, my system finally should have this type of box (Refer Slide Time: 22:08); one input X, one output z, one clock. The edge triggered clock I put it like this. This is the box which is the unknown system which we designed using this X corresponds to this input. This z corresponds to this output and this clock is this clock. Otherwise, the 0s and 1s, 1 is a V_{CC}, 0 is a ground and output comes here into Flip-flop and all this is a part of the design.

This is a very simple design for two reasons. The first reason is, this part of the circuit is same (Refer Slide Time: 23:00). Any circuit or design up to 4 states will have the same 2 multiplexer architectures. The only things that will be different will be these inputs to this multiplexer. For the given architecture with 2 Flip-flops and 2 multiplexers, I can design any 4 state machines. Of course the combinational logic output will also be different. The second thing is, the design is so simple I do not have to go through karnaugh map reduction etc. All I have to do is to direct mapping. I have the state table, the hardware, so I match the state table to the inputs of this Flip-flop. That is why it is called an MSI-based design. MSI stands for Medium Scale Integration circuit, as I already told you at the beginning of one of these lectures.

These 2, 4 to 1 multiplexers can be obtained as a single package. We have one package for this and if you want two separate, you can get 2 flip-flops. Otherwise, you can have a duel flip-flop, which is also a package; so one package for this and one for this and one AND gate, one inverter that will be the part count for this problem.

(Refer Slide Tim: 24:23)

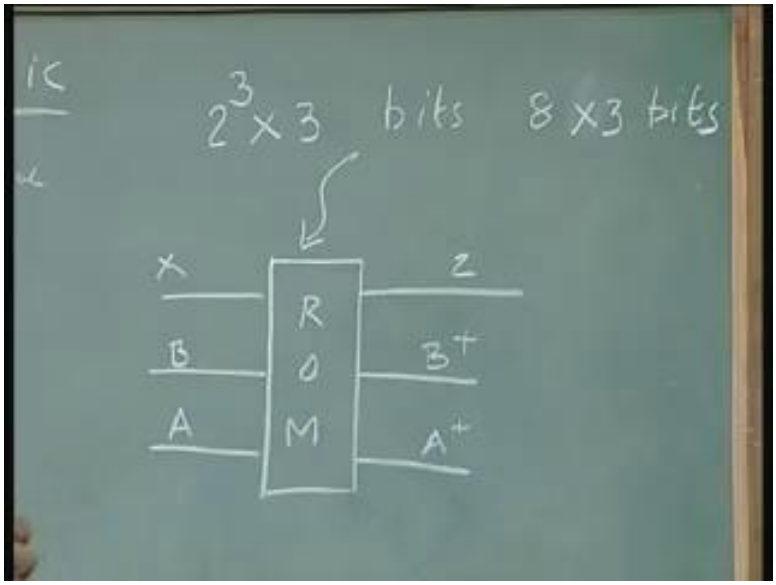


Let us move on; we can make it even simpler. The next thing we saw was ROM based design. Can we use ROM instead of this MUX-based combinational logic? Let me replace the mux by ROM, which is easier because you know that ROM is nothing but a hard wired truth table and the truth table is nothing but the state table. I am going to use the same truth table in a present state, input, next state, output and the same table where this is A B X A plus B Plus z; 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 that is state table written as a truth table (Refer Slide Time: 25:54); the information is the same. In the case of a state table, the information is that the state variable changes based on the present state of the input and that is why it is called the state table. When I call it a truth table, it is an input combination versus output combination but the information is same in both cases. The output out of the truth table would be 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 0.

Now, what I am trying to do is, I am going to take this ROM as input combination logic and replace MUX using a ROM. Earlier, I replaced gates with MUX, now I am going to replace MUX with ROM. What the size of this ROM? There are only 3 input variables namely; the state

variable and input; for each of these, there must be a corresponding next state information and the output information; so, I need to have 8 locations in this ROM.

(Refer slide Time: 26:58)



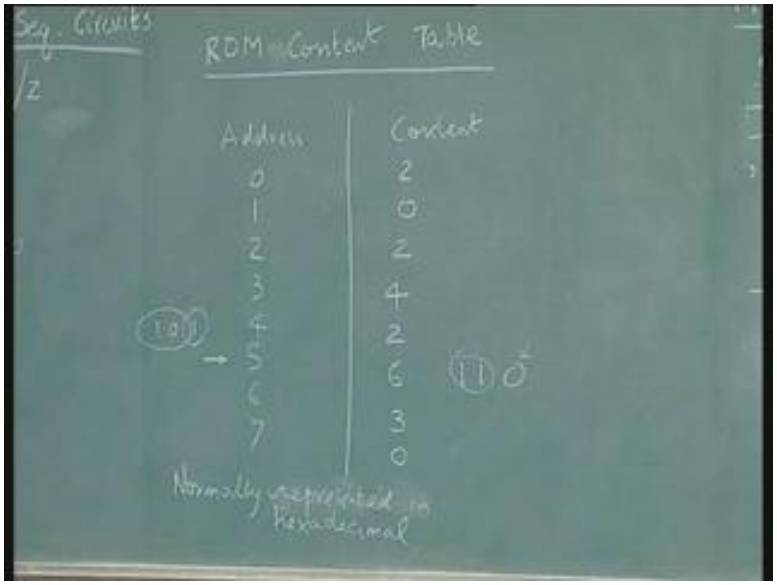
There are 3 inputs and 8 outputs in this ROM so it is a 3 address ROM. One of them would be X B A. These are the 3 variables to the ROM input variables. The output of ROM would be z B plus and A plus and this is ROM (Refer Slide Time: 28:04), the size of which is 2 to the power 3. There are 8 words 3 inputs 2 to the power 3 and each word as a 3 bit output. 2 power 3 by 8 or so many bits or 8 into 3 bits. Do not multiply this and say 24 bits because, if I multiply this and say 24 bits, you do not know the organization of the ROM. I need to infer the best thing is 2 to the power 3 into 3; then 2 to the power 3 is 8 so you can say that 8 times 3. I want ROM with 8 locations each of 3 bits width; that is what it means. You multiply and say it is 24 bit ROM, you do not know how many words, what is the width, we do not know.

What happens to the output? It is fed back into registers as inputs; these are D inputs Q outputs - a ROM based design. This whole thing is just to program this table which is a state table into a hardware and write it into this ROM. People when they do a ROM design, do not usually give this table because, ROM is usually used for designs which are very large but we have only 8 words, so you can afford to have this table in this form, in binary form.

Suppose I do a very huge design, there are several inputs and several outputs. If you write the whole thing, the binary table becomes so large. So, normally, ROM table is expressed either is a

decimal or a hexadecimal information. Each of these are called an address so ROM address (Refer Slide Time: 30:33) 0 0 0 0 0 1 0 1 0 so these are the call addresses and these are called word content. What is the address in corresponding content of the word in that address? That is what it is; it will give an address; so it is called ROM table. This is not called state table, not called a truth table. They call it ROM table in which they will give an address on the left hand side and content of the address. The content is the ROM content, word content or ROM content so it is going to look something like this.

(Refer Slide Time 31:33)



We have to draw in a ROM design, what is known as ROM content table. The design is not complete until you give a ROM content table. You give an address and content of that address. It is not a big deal; all you do is write it in the form of a concatenation. Now it is a binary 0 0 0 0 1 0 etc., concatenate them to give it as a word. For example, this case 0, 7 address are there, and each of the addresses, first content as 2, the second content as 0, the third content is 2, the fourth content is 4 then it is 2 again, 6, 3 and 0 (Refer Slide Time: 32:38). This is the content that is the address location 0 contains information 2. When you decode this information 2 into a binary, it becomes 0 1 0 that is what you want 0 1 0.

For example, location 4 content is 2. Let us take some other content, location 5 has the content 6. 5 is 1 0 1 and that means it corresponds to the 10 state and the input 1 because this is expanded to become 1 0 1. The first two are state variables and the next is input variable. When state is 10 and

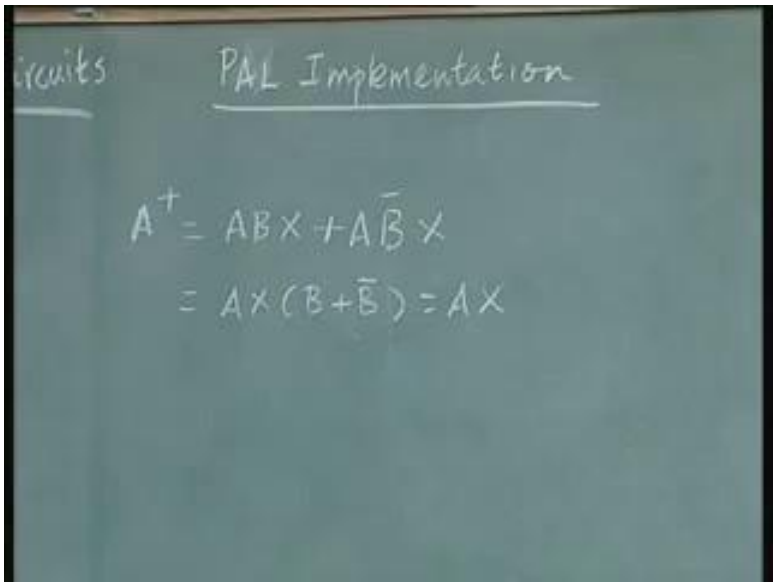
the input is 1, what should be output? 110 means the next state is 11 and the output is z. All this information is contained in this ROM table. This is a very simple, trivial example; usually they have many more rows and they use a hexadecimal representation. They do not stop at the decimal; it is normally represented in hexadecimal.

The simplification is obvious; from gates to multiplexer there is reduction of hardware. From the multiplexer to ROM there is a reduction of hardware because, the whole ROM table is one piece with no invertors (Refer Slide time: 34:17). In the multiplexer, it had inverter X bar and all that is not there and extra gates for the output was not there. There is an output z gate in multiplexer design which is not in there in this case. ROM of the correct size is followed by 2 registers and again 2 Flip-flops which can be given as one single IC. Now I have reduced further. In the next level of simplification of hardware, that means integrating in the hardware. I am reducing the hardware count from the gate to multiplexer and from multiplexer to ROM. Even the ROM designed we had ROM and a register which is 2 Flip-flops.

Now can we integrate The Flip-flop also into the hardware? Yes, it is possible because you have already seen these different versions of the programmable logic devices PLAs and PALs. This PAL also have registered outputs, we saw when we discussed PALs we said that there is also the possibility to have these programmable logic devices PALs with registers in the output. That means, the sequential circuit requirement of a Flip-flop can also be built into the PAL. So with a single PAL I can now get the whole circuit in combination as well as the combination logic and the registers or Flip-flops. For that, again let us look at the truth table. (Refer Slide Time: 35:42) We will go through the PAL implementation again and the same state graph I am going to implement using programmable array logic; earlier multiplexer based and ROM based only replaced the combinational logic.

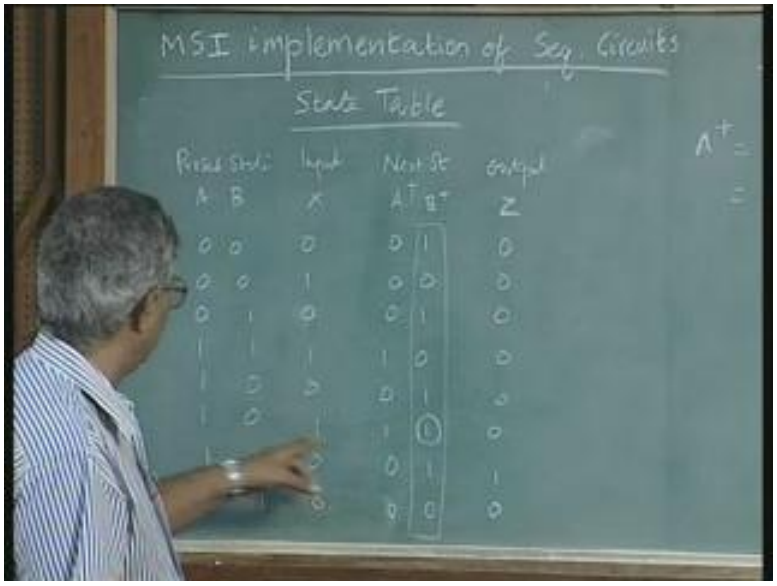
The PAL is going to replace the whole thing; there is no combinational logic, no sequential logic, nothing. Everything will be put in one chip then the design is completed. For that let us look at these state tables one more time. (Refer Slide time: 36:10) This is the same table we are using again and again. Now what are the things, I need a next state value for A plus so that it will be given to another Flip-flops A, next state value for B plus which is given to Flip-flop B in the output. These are 3 things I am interested in, A plus B plus and C are the outputs.

(Refer slide Time: 36:43)



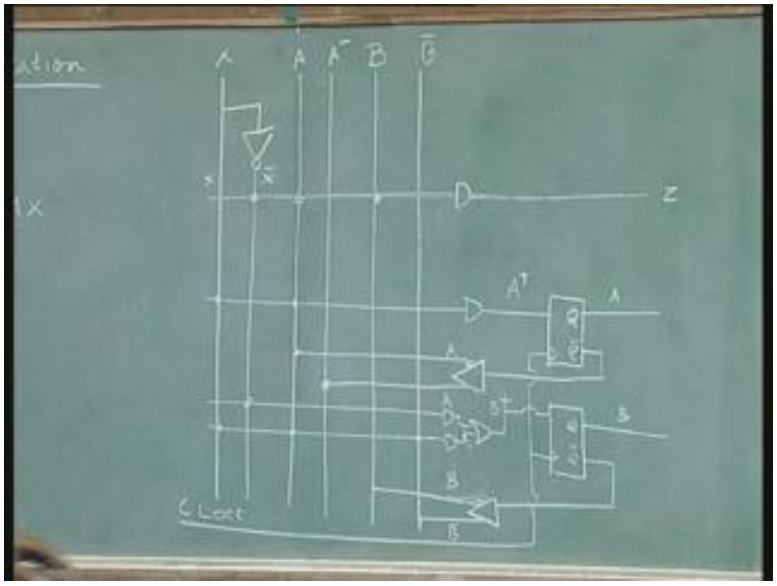
Now A plus, this is a combinational logic input 0 A B X are the 3 inputs and A plus is the output. So you can do a karnaugh map or whatever you want or you can just by inspection you write that A plus is 1 when A B X all are 1. So A plus is A B X, or A B bar X, which is I can take A X (B plus B bar) is equal to A X. So, A plus is nothing but A X. So If I can get this term A X in my programmable array logic you know that the programmable array logic PAL has a combinational part where I can combine the inputs to form any output and also have a sequential part, I can store it in the register. So in one of the combinational gates we can get AX, I can get A plus.

(Refer Slide Time: 38:31)



Now locate B plus and the second output required is a next state variable B which is B plus. Look at these patterns, very interesting pattern B plus is wherever X is 0, B plus is 1, 00 1, 01 again 1, 10 again 1, 11 again 1; whenever X is 0, I get B plus 1. The extra term here is a 10 X. So my (Refer Slide Time: 38:38) B plus would be X bar. This is the B plus term and the output term z is again A B X bar we had seen this before. If we can get these 3 terms implemented in the combinational logic part of the programmable array logic and have 2 Flip-flops to store these A plus value and B plus values and feed them back as A and B in the next clock cycle then I have to implement the whole thing. I know fortunately we have Flip-flops; I mean, PAL's with combinational inputs gate structure, Flip-flops of the output and also the facility of feeding the Flip-flop output back to the input.

(Refer slide Time: 39:40)



Let us look at a schematic of a one such PAL - this is not any specific device type number. Usually we go for a device type number. I am giving you the sort of requirement; practically available PALs or even more complex will have more features; in fact, more combinational inputs, more registers. But, I am giving a minimum configuration required for this. This is not any practical part available, it is a minimum requirement; sort of a theoretical assumption.

I am assuming I have a PAL of these and it is not an assumption which is not valid because this as I said is one of the low end requirement; whereas even the smallest PAL available in the market will have more features and more capabilities. That way I will not be doing anything

wrong assuming this is available. We have this A X given to this and X and X bar of course. This is X bar. I have 2 Flip-flops, there are many registers in the output I am going to choose two of them, one of them Q and Q bar a Q bar is fed back. This becomes A bar, so I put an inverter for A and B (Refer Slide Time: 41:05). These are all the inputs to the combination logic X A B C. (Refer Slide Time: 41:12) the combination logic here I need is X A B so I should AND X bar A bar B bar.

Since all of them are required, I need to create all of them here: X A bar A A bar. Similarly, the second Flip-flop will have output B derived here (Refer Slide Time: 41:30). The second Flip-flop B and this should be B bar fed here and this is a clock though this Flip-flop. Now all I have to do is to create this term for A plus, create the term for B plus and create the term for z. Accordingly these equations using the combinational gates available, z is very easy is A B X bar - it is A B X bar so we have AND gate.

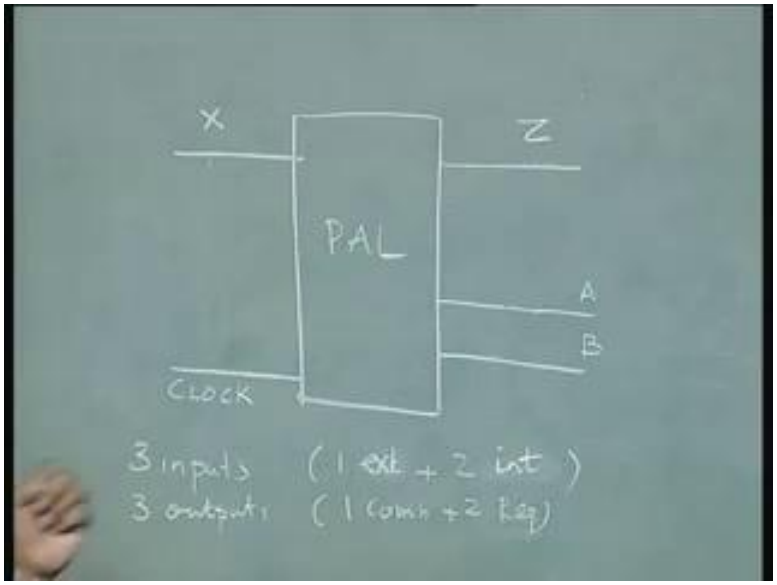
I am not going to show the actual way in which this is drawn in the book - this is enough. We have a combinational gate. Of course, there are several AND gates feeding to an OR gate and the program such that you get the output. This is only a symbolic representation. I am going to combine the AND gate possibly; so I am going to combine this and this and this and get this requirement (Refer Slide Time: 42:27). This is only one of the several terms you can put into OR gate actually in a practical device. There will be an OR gate output and the OR gate will have several inputs, each of the inputs is an AND gate combination.

Coming to the second, A plus, there is again every simple is the A X bar. I am going to have another AND gate to use for this purpose. A X this is my A plus which becomes A after the clock. The next state value is present state value and finally B plus have 2 terms X bar or A B bar X. So I need 2 AND gates combine into an OR gate; 1 AND gate, other AND gate. This is only a symbolic representation as I said; the practical circuit will have more AND gates feeding to every OR gate, so you have more flexibility and more possibilities.

This is my B plus it is B what are the 2 times required for that? One is simply X bar, so let us have this X bar as the one of the terms, one of the inputs to the OR gate; the other input is OR gate is A B bar X which is A B bar X. So A B bar X or X bar together gives to the next state variable B plus; A and X gives the next state variable A plus; X bar A B gives the next state

output variable z. So in one IC which is a programmable array logic IC, having the proper inputs as given, we can realize the whole circuits.

(Refer Slide Time: 45:10)



I have here one PAL in which I will have the following inputs X, z as the output; nothing else because, this A going back as input, B going back these are internals. So, I have the outputs of A and B required. Of course, you do not have the required in this case so the register outputs are available but we are not going to use them and a clock. These are the only inputs I have used. The requirement of this PAL is the minimum requirement; this is a very, very simple, trivial PAL; any standard PAL available in the market will have many more features. So I need 3 inputs: this 1 external input and 2 internal inputs. The convention is to call it as 3 inputs because I should be able to connect up to 3 inputs whether they are internal and external. Even though it is one external input they usually call 3 inputs and only 1 output is required. But then again, A and B are not required but they are available; so, I should count them also as 3 outputs.

Again, 1 combinational plus 2 registers (Refer Slide Time: 46:40); this will have the minimum requirement of this would be, if you define if somebody asks you to give the specifications requirement for this design to be carried out a single PAL, you do not know what is available. Suppose you do not know what is available in the market and somebody asks you to do a minimum specification for the design, you will say give me a PAL with 3 inputs. One of them

external, 2 internal. I have my students watching here who are not informing me that I have made a mistake. 1 external and 2 internal - 3 outputs.

One of the combinational output for the output z other 2 registers for this is for X input, this is for A plus B p plus (Refer Slide Time: 47:39). Similarly, one combinational is for z, 2 required for A plus B plus, 2 registers and then will have to say at least enough number of OR gates enough number of AND gates for each of these OR gates. This is a trivial requirement because, I need only 1 AND gate for the X, for the z I need only 1 AND gate, for B I need 2 AND gates. I need again 1 AND gate. So, I need at least 1 OR gate with 2 inputs and I do not have to have any other OR gate, 4 AND gates should be there, out of these 2 of these two of the AND gates should be together to able to given to OR gate.

These are the minimum requirements. Of course, this should be clock input to be able to give clock also it should have power supply V_{CC} and ground. This is as I said just trying to go for the other end, I took a very simple problem. In one lecture and I want to finish the complete design flow from the gate level to multiplexer level to ROM to PAL, to show you the gradual reduction in the hardware requirement. Less and less number of pieces of hardware, circuit will become more and more efficient, reliable and easier to implement. That is the purpose of this lecture; that is why the trivial example was taken; but in practice, this PAL will have much more capability.

For example, the standard PAL will be at least having 8 inputs combinational - external inputs 8 outputs. Out of them 4 of them may be combinational output, 4 of them may be register and these register output can also be brought back as inputs. We will have 8 external inputs 4 internal inputs as a typical scenario. 4 of them combinational, 4 of them sequential, a clocking and each of these outputs, whether combinational or registered outputs, will have OR gate inputs which can take several AND gates as inputs to the OR gate.

Lots of these things can be found; you can have much bigger, more complex design of systems using a single PAL. I am just giving the gist to you so that in one single IC I do not have any registers, I do not have inverters, I do not have gates, nothing. PAL I buy and program it for whatever I want; my block box is here (Refer Slide Time: 50:33). This input, output, clock, this is my original circuit, this is my pattern detector and I need 1 input, 1 output and one clock. In a single IC, I have a whole system. In the next lecture, we will take a bigger design and see how we can fit into an available PAL. We will stop today. Thank you.

Summary of Lecture 8

(Refer Slide Time 51:06)

Next Lecture:

Design of Sequential Circuits using One Hot Controller

(Refer Slide Time: 51:51)