Digital Circuits and Systems

Prof. S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology, Madras

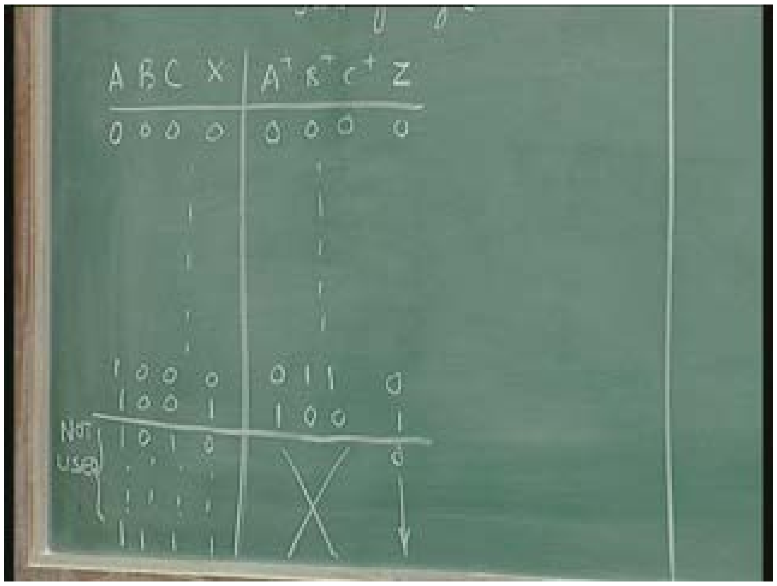
Lecture - 35

MSI and LSI based implementation of sequential circuits (contd...)

So we were discussing the implementation of the sequential circuits using MSI and LSI components. We saw how to do a design of sequential circuit with multiplexers as the combinational logic which is used to drive the flip-flops. We also did the rom implementation prom implementation rather, for which you write the transition table or the state table and get the size of the rom and program the rom to write the contents of the table so it's called the rom content table. The same table which is what you call as the truth table for combinational logic then programmed into a rom then it is called a rom content table. So the input variable is called addresses and the output variables are called the contents in the addresses so that is why it is called rom contents table.

Then we talked about the Programmable Logic Array solutions wherein before we decide the size of the PAL or PLA we need to know how many min terms how many product terms we need to accommodate so we need to reduce the hardware the table the state table and the transition table to as few product terms as possible. When you do a Programmable Logic Array implementation we would like to have as many common product terms as possible whereas in the case of Programmable Array Logic commonality of the product terms is not going to have any meaning because it does not help. It does not help you to reduce the size because the OR gate inputs are prefixed. So the same state graph that we discussed in the last lecturer we will use as an example for implementing PLA Programmable Logic Array and PAL Programmable Array Logic solutions.

(Refer Slide Time: 5:20)

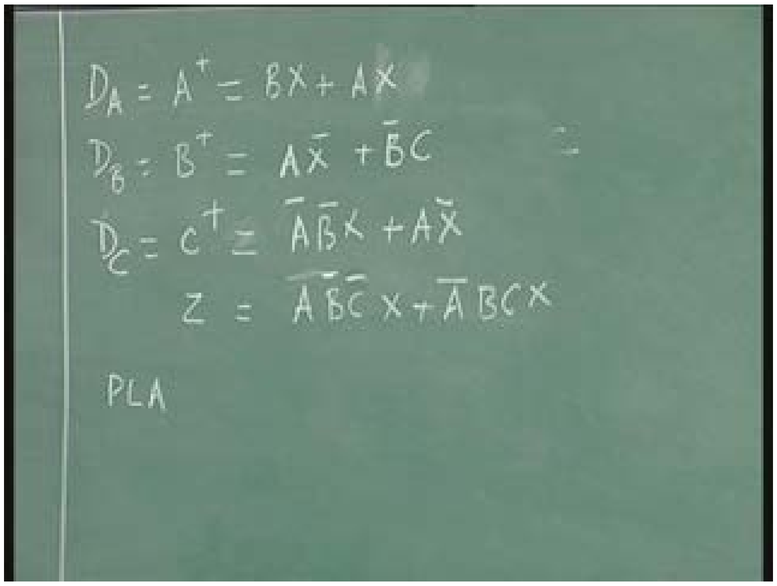


	A	B	C	X	A+	B+	C+	Z
	0	0	0	0	0	0	0	0
	1	0	0	0	0	1	1	0
	1	0	0	1	1	0	0	1
Not Used	1	0	1	0	X	X	X	0
	1	1	1	1				

Without going to details of the simplification which you know how to do, now we will see…….we had this truth table, I am not going to write the full truth table because we did that in the last lecture A B C are the present state variables along with input X and A plus B plus C plus are the next state variables along with output Z and for this table the values are (Refer Slide Time: 5:55) these, rest was given to you earlier, the same table and these states are not going to occur so this will be filled with don't cares and these will be filled with 0s and this part shown here is not used in this example (Refer Slide Time: 6:42).

So on simplification I get the following D_A which is same as A plus the next state variable, A plus is same as the input to the D flip-flop as BX plus AX then D_B which is the next state variable B plus also the input B flip-flops. So, for D_B it is B plus is equal to AX bar plus B bar C then for D_C it is equal to C plus is equal to A bar B bar X plus AX bar which is a common term here. Then we have Z the output equal to AB bar C bar X plus A bar B CX.

(Refer Slide Time: 12: 30)



$D_A = A^+ = BX + AX$

$D_B = B^+ = A\bar{X} + \bar{B}C$

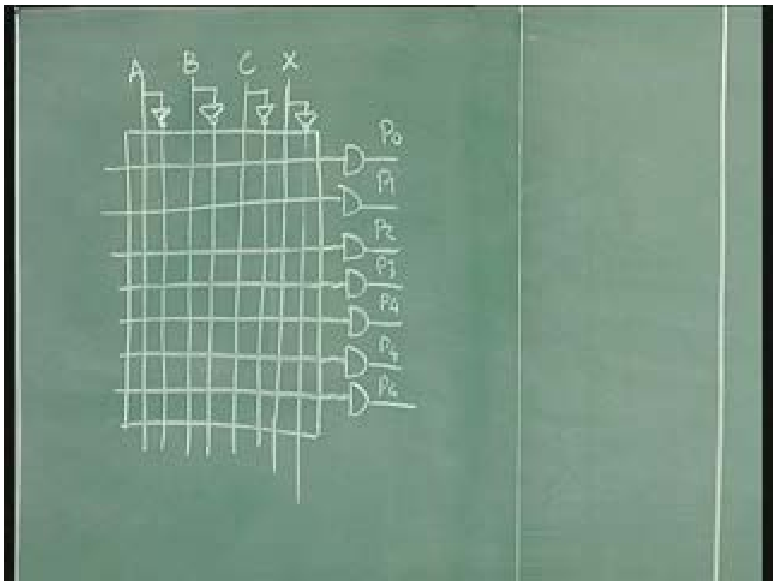
$D_C = C^+ = \bar{A}\bar{B}X + A\bar{X}$

$Z = \bar{A}\bar{B}\bar{C}X + \bar{A}BCX$

PLA

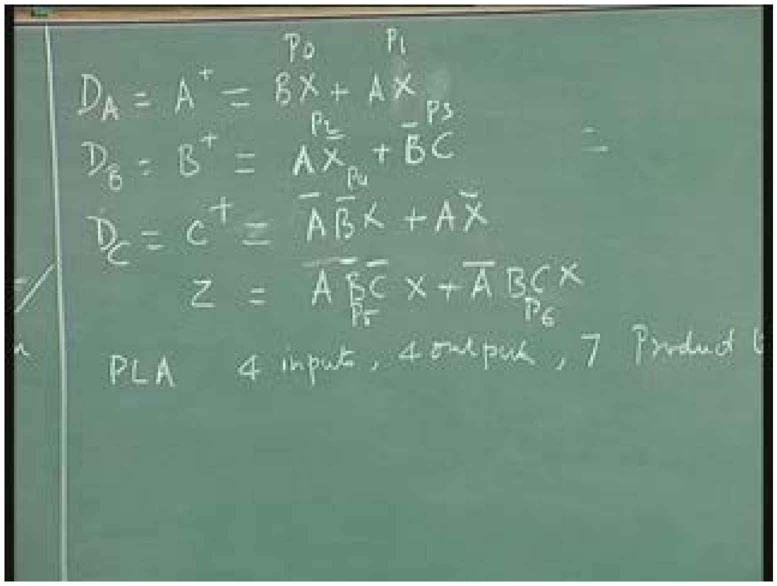
So we have to now give the size of the PLA, this you get by simplifying this graph from this table (Refer Slide Time: 12:40) which is a state table. A state table is also a transition table in this case because of the fact that we are using D flip-flops. If you use some other table the transition table will be slightly different. So the size of the Programmable Logic Array is, how many inputs, four inputs four outputs and how many different product terms are there now? There are 1 2 3 4 5 6 7 so 7 product terms so we can get IC with four inputs four outputs and 7 product terms and do this programming. The way it is done is A A bar B B bar C C bar X X bar so each of this may be inputs to produce these seven required product terms so I need to run lines. This is the symbol to show that any of these can be input to the AND gates.

(Refer Slide Time: 15: 40)



That means these AND gates have eight inputs each, some of them you can retain and some you can disconnect so like that I need seven terms 2 3 4 5 6 7 so this will be called P_1 P_2 P_3 P_4 P_5 P_6 P_7 and this (Refer Slide Time: 16:00) is my P_1 P_2 P_3 P_4 and this is also P_2 P_5 P_6 so 7 product terms. BX is the first one so it will be B and X, second is AX, third will be B bar C, P_4 is A bar B bar X P_2 is AX bar; P_3 is B bar C; P_4 is A bar B bar X; P_5 is A B bar C bar X; P_6 is A bar B C X.

(Refer Slide Time: 16:50)



$$D_A = A^+ = \overset{P_0}{BX} + \overset{P_1}{AX}$$

$$D_B = B^+ = \overset{P_2}{A\bar{X}} + \overset{P_3}{\bar{B}C}$$

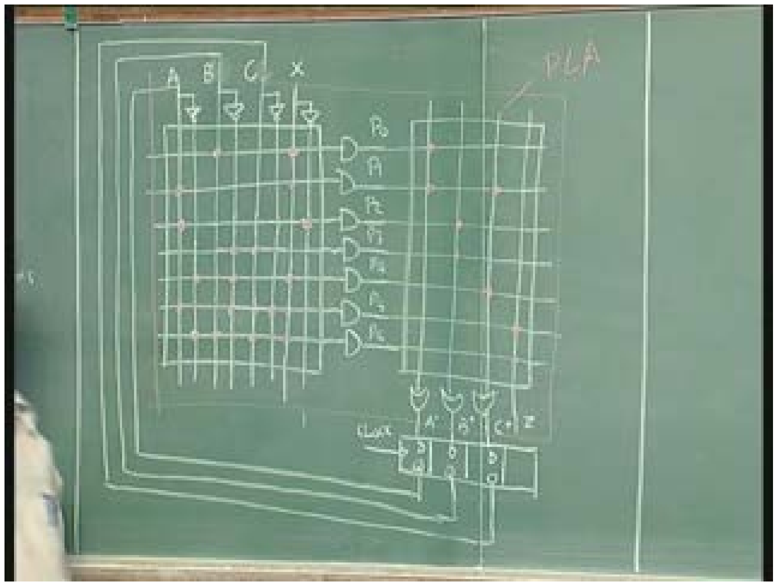
$$D_C = C^+ = \overset{P_4}{\bar{A}\bar{B}X} + A\bar{X}$$

$$Z = \underset{P_5}{A\bar{B}\bar{C}X} + \underset{P_6}{\bar{A}BCX}$$

PLA 4 input, 4 output, 7 Product

So this is (Refer Slide Time: 20:00) P_0 P_1 P_2 P_3 P_4 P_5 P_6. Now I have to combine them in OR gates which is a symbolic way of showing this. This is A plus B plus so P_0 P_1 P_2 up to P_3 P_4 P_2 P_5 P_6 so this is the implementation. Now you are familiar with the drawing, three flip-flops so the next state variables are stored in the flip-flops are used as the present state variables this completes our design; Programmable Logic Array based design of the same state graph. This whole thing is a single IC and this is the PLA one four bit register, there is nothing like a three bit register.

(Refer Slide Time: 23:00)

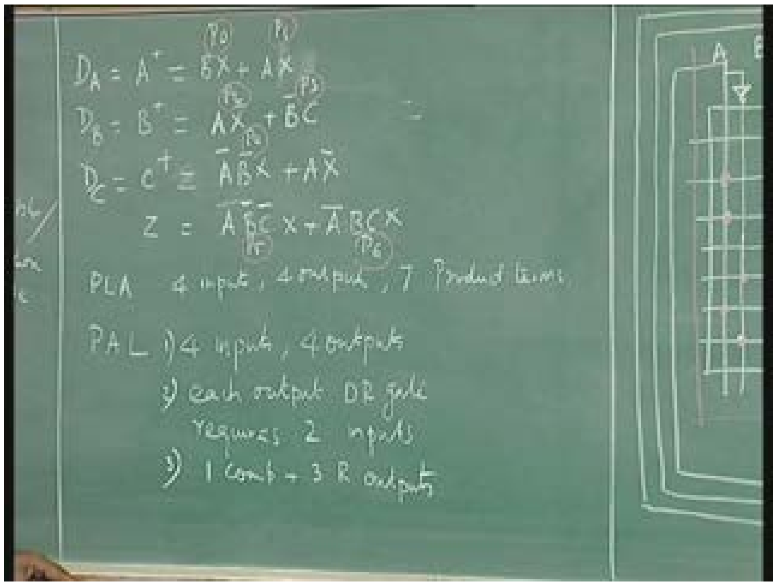


PAL solution is similar, this is a PLA, there is a change here (Refer Slide Time: 24:08) C plus is P_4 P_2, the dots are not important finally when you make corrections in programming.

For a PAL solution, I need four inputs, four outputs and there is no point in mentioning the 7 product terms because we need to give the number of inputs to each of these OR gates, each of these outputs require two inputs so each output OR gate requires two inputs. Then if possible I would like to have registered outputs for at least three registers. I told you output can be combinational output, high low, exclusive OR, programmable I/O so I can have a registered output, one combinational and three registered outputs minimum.

Of course you will not get all of them, if you search in the website of the manufacture of this IC PALs and you give this requirement it might come up with a list of ICs which satisfies the minimum conditions but they may not have exactly the same number. It may be very difficult to get an IC with three registered outputs may be one with a four, four registered and four combinational will be possible, four registered and two combinational may be possible, four inputs may not be possible may be four is possible, may be eight inputs so it is something like that. So I am not going to do that.

(Refer Slide Time: 26:00)



$D_A = A^+ = \overset{P_0}{\bar{B}X} + \overset{P_1}{A\bar{X}}$

$D_B = B^+ = \overset{P_2}{A\bar{X}} + \overset{P_3}{\bar{B}C}$

$D_C = C^+ = \bar{A}\bar{B}X + \overset{P_4}{A\bar{X}}$

$Z = \underset{P_5}{\bar{A}\bar{B}\bar{C}X} + \underset{P_6}{\bar{A}BCX}$

PLA 4 input, 4 output, 7 Product terms

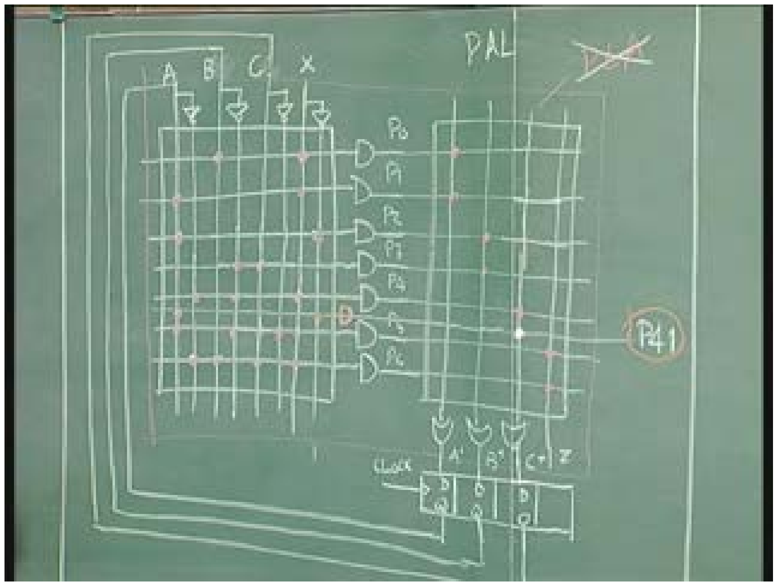
PAL 1) 4 input, 4 outputs

2) each output OR gate requires 2 inputs

3) 1 Comb + 3 R outputs

As I said there are some standard ICs the drawings of some of them are given in standard text books, you see how it is connected. So the whole thing is, without redoing the whole exercise what I will do is, to make it as a PAL solution………., this is a PLA solution (Refer Slide Time: 27:09) suppose I want to make it as a PAL solution, this part is same, this part is same except that you need to get one more input so this input we will call it P_3 P_4 this should be P_5 P_6 that means this AX bar will be generated twice; once it will be required for this output again it will be required for this output so in order to modify this graph this diagram I am going to call this P41 just like that. Let us call this product term P41 product term that is modification of P_4, same as P_2 it is, that means I need to introduce it after P_3 P_4 before P_5 somewhere here (Refer Slide Time: 28:17).

(Refer Slide Time: 28:19)



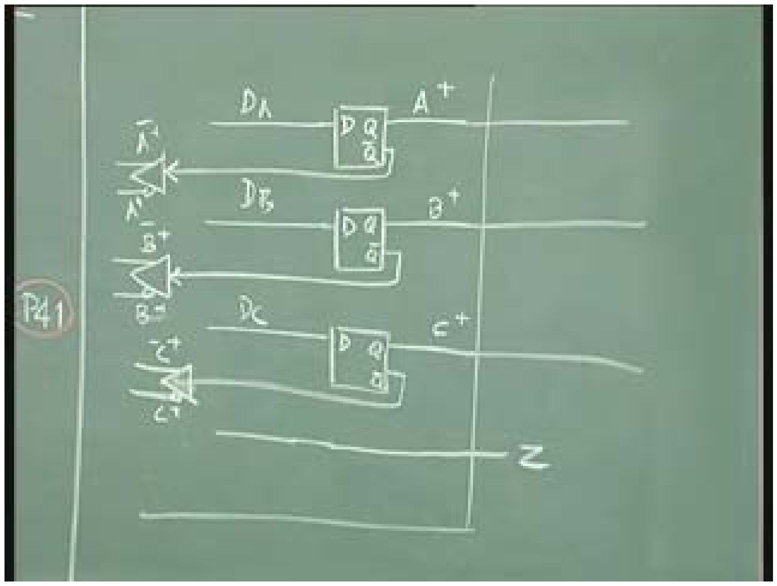
This is the modified product term P41 which will have again AX bar same as P_2 with AND gate. So these two terms will go into my A plus; these two terms will go into B plus, these two terms will go into these and this and this will go into………

Now I have to modify this graph, remove this input because this is not available, introduce this as extra. Now this term and this term (Refer Slide Time: 29:55) together will go to C plus and Z is same. So this term has to be generated, this is what I wanted to say. Even though it is a common term between C plus and B plus I need to generate it two times; once into the B plus output and second into the C plus output because these outputs are fixed into the input of the OR gates. There is no way I can manipulate it, it is not programmable it's fixed. This is programmable (Refer Slide Time: 30:30) how we generate P_0 P_1 P_2 P_3 P_4 P_5 P_6 is in our hands but what product terms are connected to each of these outputs is not programmable, is not negotiable, it is fixed.

Thus, I need to have a device with at least two inputs for each of these outputs; first two inputs go for this; second two inputs go for this; third input goes for this and so on. So I need to generate this. Of course fortunately I will have this extra term generated and again these registers need not be outside this to be a part of this, only that part I will show you here. So what it can be?
Supposing your outputs are DA DB DC and Z four outputs are generated, these are the points, each of them will go into a flip-flop in this case; D flip-flop, D flip-flop, D flip-flop and this is the Z (Refer Slide Time: 32:16). This is my A B C as the outputs and for the next state variable this has to go back into the inputs.

Hence, instead of directly connecting this (Refer Slide Time: 33:04) through the register this feedback connection is also inside the device, this feedback connection was external to the device, remember this was the device boundary for PLA and the feedback connection is external.

(Refer Slide Time: 34:10)



In this case I have the registers inside so this feedback mechanism is also inside there again inversion has to be done so now we will have…… so this is DC, this is C plus, this has to be C plus, and this is C plus bar so this will be C plus bar and this is C plus, B plus bar B plus… so going back, so this (Refer Slide Time: 34:27) is my device boundary here. I do not need these outputs I only need Z in this case, the single input single output system, X is the input Z the output, but if you want I can always look at what is in A B C and monitor them.

Limitation of the Programmable Array Logic selection is that we need extra product terms. Even the product terms that are generated earlier cannot be used, I need to generate it each time. If this is part of an output function even though the same term has appeared as a part of an output function earlier, we cannot use it.

I need not combine this, look for the common terms etc, I can always say first two terms, three terms, four terms and so on. This is just for an example. But I can give a list of two outputs with two inputs product terms, two outputs with three product terms as inputs, two with four product terms with inputs and so on. So all that requirement I can give and then we can see whether the manufacturer has a device like that. If the product does not have it then we have to go for a bigger device or go for another manufacturer.

Occasionally you can do a trick. Suppose I have three terms here C plus Z three terms suppose of course we have reduced it, suppose I had three product terms and I have this Programmable Array Logic with two inputs each I got this four outputs two input each OR gate and I have two choices now, I can make use of this and try to get all the terms, I will have no problem with A plus B plus and Z but C plus I cannot accommodate because C plus has three product terms and I don’t know where to put the third product term.

One solution is to get a bigger device with two two three three or something like that. Suppose I don't want to do that then there are extra outputs that are available because I will not be using all the outputs. For example, there is nothing like a four input four output device. Usually it is eight input eight outputs, even out of those eight outputs you may have four of them registered and four of them combinational so typically you will have eight inputs four registered outputs four non-registered outputs and I need only one non-registered output Z.

I need only three registered outputs A B C and one non-registered output Z so there are three non-registered outputs available for free which are not used and one registered output which is not used freely. What I will do is I will generate an extra term. Suppose I have P_7, for this solution this is the best one for this implementation, assume my implementation is different, assume my requirement is different, assume that the state table and state graph is different I would not be able to reduce C plus beyond the three terms so a P_7 is not generated.

So what I will do is, I will generate P_7 as an extra combinational output and feed it back as one of the inputs so that I can combine P_7 with P_4 and P41 in order to get the DC.

What I am saying is this is my requirement, this is not used, similarly this is Z. Supposing I use this to get a P_7 a single input OR gate, I have a two input OR gate in which one input I will connect P_7 and to the other input I will not connect anything so I will connect P_7 here.

Now I can get this P_7 and feed it to one of the non-used inputs, so in addition to A B C and X I also have P_7 as my inputs. If I am using eight of them I only have four unused inputs and to one of those unused inputs I am going to connect P_7. Therefore now all my combinations can be A B C X and P_7 so P_4 I will get from A B C X, P41 I will get from A B C X, and we have P_7 so I combine them together to form one AND gate product and that will be my C plus.

Hence, I can breakdown my design with unused outputs and unused inputs to avoid going for a bigger IC. Many times we are asked to do that especially in the exams. you may have an extra input and extra output which you may not want to leave waste and then go for a bigger IC you have to optimize it, if all else fails. One of the combination output is an extra term I am not able to get the….. but I tried my best to accommodate an unused input and unused output but I am not able to do anything. The best thing is to get the output with one term less and build a simple AND gate output outside of this PAL, …………. the PAL I can also generate any product term using gates. It is not that product terms cannot be generated outside gates, product terms can be generated as long as there are gates. If I am given AND gates and inverters I can always generate product terms.

So I will generate a product term or a two outside and then try to combine with the available output of the PAL and come up with a required product outputs. These are done only in case of a small extra requirement. Generally it is better to go for a bigger device. If you are talking of very very marginal shortfall in the available device or in the design requirement so the choice of using this device or a bigger device, as I said twenty pins and the next thing is forty pins and just for the sake of one product term [……..I will lose…………….42:07].

But I am talking of many many extra product terms and this device is too small to accommodate my requirement so will always go for a bigger device. So the decision is reversed cost-wise convenience-wise, the number of ICs wise because after all the number of ICs have to be reduced, kept to a small value space PCB, the circuit board, PCB stands for Printer Circuit Board, the space for the IC in the PCB, the cost of the ICs, the reliability of having a largest circuit compared to many small circuits all those power requirements and all that we said like speed requirements will be met by bigger ICs but occasionally we may have a very very marginal extra requirement, very trivial small requirement that is extra and the next choice will only double the size which will consume lot of power and be too expensive or you may not have that version available.

Sometimes this also happens with the requirement of the inventory. Suppose you have only this and the other device is not available for some reason, when the model or version is discontinued or because your company wants to use a product of a specific type particular manufacturer because of a tie up, these are all commercial problems, your company may not want to use the product of a competitor, the rival in the field so … [.. 44:02] all these are practical problems.

Many times it is possible in a simple design like this, small designs like this, many times it is possible to do a little improvision intelligently, look for certain things the way in which you can overcome the difficulty. Instead of saying I require three product terms for one of the inputs and I have only all my outputs which are only two product terms so I cannot do this same thing and go to the next problem, this is not the right approach, and design engineers always be innovative.

As I said you are paid for innovation and not for making the circuit, building it and then simulating it which is a standard procedure. Of course you need to know it but most of the salary you get for drawing the best possible design, making the best possible design, using the best components both in terms of the requirement and specification but you meet this specification then do it most efficiently. So this will conclude our discussion on LSI MSI components and use of them in combinational logic as well as sequential logic but sometimes in sequential logic also there are MSIs and LSIs. The MSI and LSI we discussed so far are multiplexers, PALs and Programmalbe Logic Devices, decoders etc and combinational MSIs and LSIs.

A register is a sequential MSI. What is an MSI? Why it is more gates than a few gates. Anything about ten gates as I said thumb rule wise, ten to hundred gates. So a four bit register, a four bit counter if you take a binary counter four flip-flops with reset, clear, all those features and clocking features, master save it is count the entire number of gates and it will be more than….. fifteen or twenty gates will be there, may be to be called as an MSI so like wise the shift registers.

Therefore, it is also possible certain times to design around the given register, the sequential logic also can be designed. We only said how to optimize the combinational logic using MSI and LSI.

I will spend one class more or one lecture more on how to use some other sequential logic building block supposing flip-flops are not available. You are given a register and asked to design a state machine you can use MSI of course gates or multiplexer type no problem.

Or you may be given a 4-bit shift registered or a 4-bit counter so use any of them and then go through the state transition as defined in the state graph with of course extra combinational logic and that combinational logic can come from either gates or from LSI and MSI. So we can have an MSI gates and simple flip-flops and then can have gates and then registers or you can have combination of both.

So you can have MSI solution for combinational logic alone, I can have MSI solution for sequential logic alone or you can have MSI LSI solution for both of them. Programmable Array Logic comes in that category of MSI LSI for both together. So we will see how to do that given a counter for example for shift register, how do implement and how do you make that counter which normally covers a particular sequence to count it in the way you like for a particular combination of states given by the state graph.

Therefore with that example we will finish our discussion on LSI and MSI and move on to some system design examples.