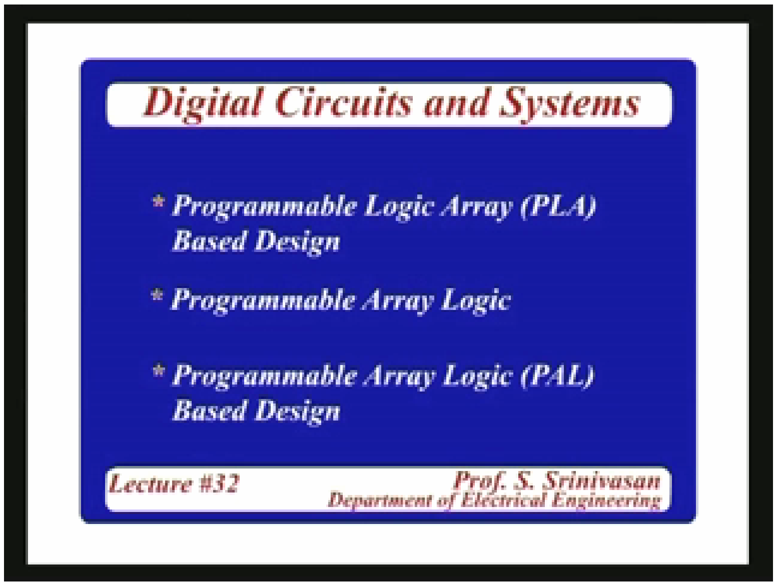
Digital Circuits and Systems

Prof. S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture - 32
Design using Programmable Logic Devices

(Refer Slide Time: 00:01:30)



Digital Circuits and Systems

* *Programmable Logic Array (PLA) Based Design*
* *Programmable Array Logic*
* *Programmable Array Logic (PAL) Based Design*

Lecture #32 Prof. S. Srinivasan Department of Electrical Engineering

We said today we will talk about Programmable Logic Arrays. We talked above programmable ROMs programmable Read Only Memory in the last lecture, the uses programmable ROMs in design of combinational logic. The next PLD Programmable Logic Device which we will consider is called Programmable Logic Array written as PLA in short. The ROM based design or the PROM based design as number inputs increases the number of min terms increases not linearly, it is the power 2, it is sort of an exponential increase.

The size of the ROM number of depends on the number of inputs, as the number of inputs increases the size increases exponentially. Of course if the number of outputs increases then only the size increases linearly because it is the width of the word you write in each location. The width of the word write each location is the output. The number of words in the ROM is the number of min terms for that particular number of inputs which will increase as 2 to the integer power of 2. So what happens is when the system is big with large number of inputs and outputs the size of the ROM becomes very large.

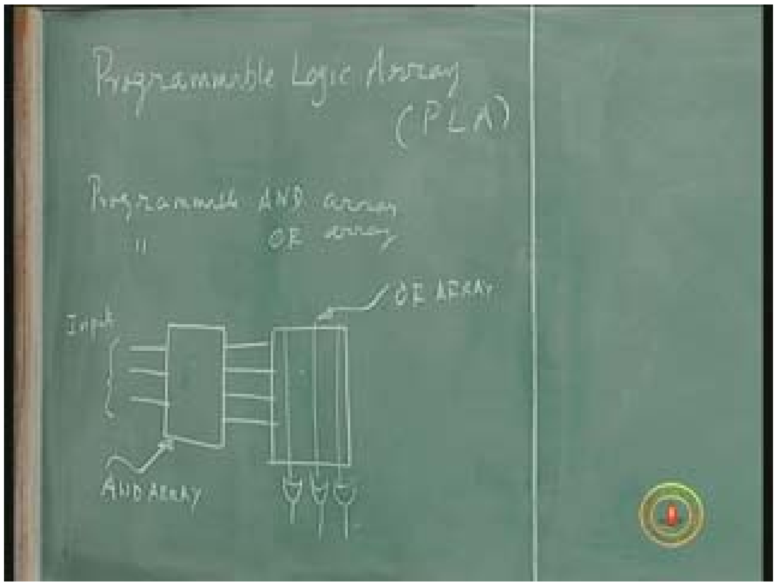
Earlier times the size was an important criteria because it was expensive to make more bigger and bigger ROMs and even now if you can save on the size of the ROM, we can say why should we unnecessarily spend silicon if you can do it in a smaller chip the size also takes a lot of space in the IC which you don't want to do because you will rather like to use that space for some other function. Anyway switching activity increases because of the power consumption and everything. That is the case for making the ROM as small as possible from the point of view of everything namely cost, power saving and so on. But then if the number of inputs is given then there is no way of reducing it. We have to generate all min terms if possible and then find out which of the min terms need to be combined for each of those outputs.

But in practice especially when the system is large with number of inputs and outputs we find that the given output is only dependant on the few min terms, very rarely it is a combination of large number of min terms. That means even though we are generating all the min terms for a given number of inputs each of the outputs will use only of the few of them. That means many min terms go unutilized or unused. That means generating more terms that required in the design so thereby came the concept the Programmable Logic Arrays. The Programmable Logic Arrays, the AND array, so remember the programmable read only memory prom based solution we had an AND array which produces all min terms and an OR array which was used to program the exact min terms that are required for a particular output.

Now, if can make the AND array also programmable, that means if you have a fixed AND array all the possible min terms if I fix a programmable AND array and programmable OR array then I can reduce the number of min terms to be generated depending on for that giving function, that given set of outputs, look at the outputs to be generated and then those min terms do not even generate other min terms meaning I can reduce the hardware. This is the concept of the Programmable Logic Array.

Hence, given as a block diagram supposing these are the number of inputs three, in the previous case in the ROM based solution we could have generated from m_0 to m_7 which would become OR array to make the difference. Here what I will do is I will generate a four, that depends on how many and what are those terms will depend on the function to be implemented, the outputs and those will be combined in the OR array. So this is programmable, this is also programmable. Earlier this was fixed and this was programmable. In this case this is programmable (Refer Slide Time: 8:00) and this is also programmable and again we will have the………

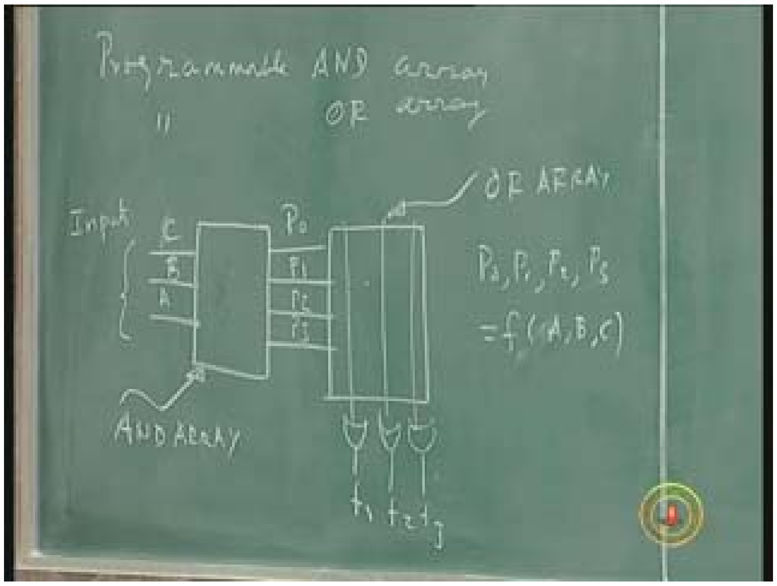
(Refer Slide Time: 8:15)



The question is I need to reduce the given function or given set of functions in order to decide which are the terms which I need to generate, it need not be min terms. Now I will have to do a simplification which was not done in the case of ROM array. The ROM implementation I said we don't have to do any simplification and all you need is a truth table. If I get a truth table for each combination whether the output is present or not if you know then you generate all those min terms anyway then tie those min terms together to make an output for a particular output. Now in this case I will do an extra step of simplification using the Karnaugh Map and generate only the smallest number of terms which are required to make all the functions possible to realize all the given output functions.

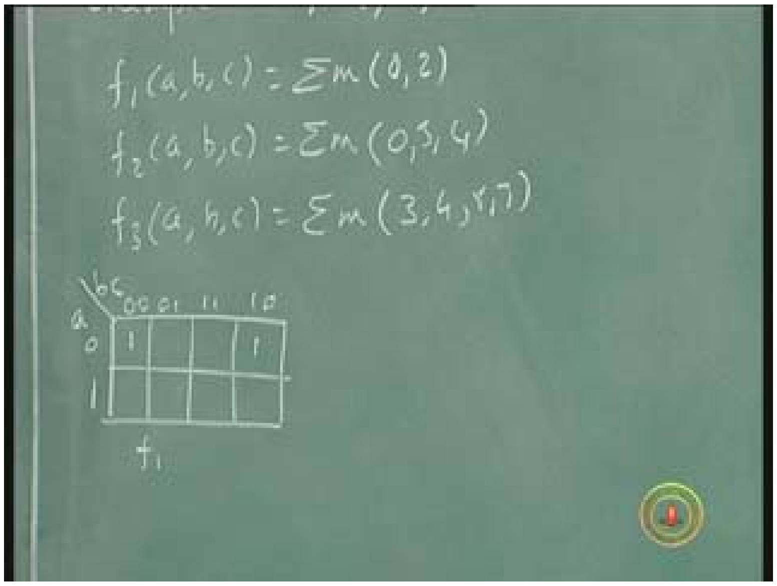
So instead of calling them m zero m one m two m three they are not min terms any more but they are product terms so I will have to call them P_0 P_1 P_2 P_3 which are again functions of the inputs so if we call this a b c as we always do then P_0 P_1 P_2 P_3 will be functions of (a, b, c) (Refer Slide Time: 9:40). Now the step is to get the specification get its truth table, the ROM implementation will directly go from the truth table to hardware and in the PLA implementation Programmable Logic Array implementation we go through a step of minimization and reduce the functions to a set of product terms which are necessary to implement those functions given f_1 f_2 f_3 f_4 but some in this case is f_1 f_2 f_3.

(Refer Slide Time: 10:20)



One more advantage is if some of those terms for f_1 f_2 products are common I need not generate it two times but I can only generate it once. In fact I can make an efficient design by having as many common terms as possible. So when simplifying f_1 f_2 f_3 Karnaugh Maps I will not have the reduction gate in mind, reduction of product terms is an important issue and at the same time I will keep in mind is it possible to have the same min terms same product terms that appear in more than one output function which means I can share the product terms of different outputs also take into account shared product terms. Let us see a simple example of three variables.

(Refer Slide Time: 12:37)

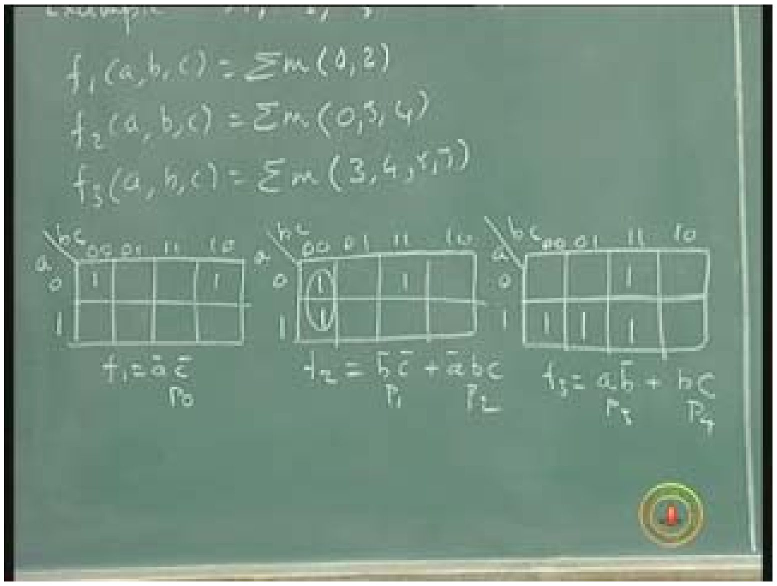


I will have three functions f_1 f_2 f_3 to be implemented with three variables (a, b, c). a truth table will be given as follows f_1 (a, b, c) is equal to sigma min term (0, 2) only two min terms are required; f_2 (a, b, c) would be sigma min terms (0, 3, 4); and f3 (a, b, c) will be sigma min terms (3, 4, 5, 7).

Now if I draw the Karnaugh Map for each one of them Karnaugh Map one f_1 (0, 1, 2) so what is this? (Refer Slide Time: 13:10) a bar c bar; f_2 would be 0 1 2 3 4 and this will be b bar c bar or a bar bc and I have a third function 0 1 2 3 4 5 this is the f_3 so this will be (Refer Slide Time: 14:50) a b bar plus bc. So how many different product terms are there implemented in all of these functions? They are 1 2 3 4 5 so I need five product terms.

Instead of generating eight min terms and combining them with (0, 2) here, (0, 3, 4) here and (3, 4, 5, 7) here I can generate five product terms because min terms require a bar b bar c bar three variables always, here this requires only two, this requires only two and this requires only two, and only here you have got three so not only I am going to use less number of AND gates AND array, less number of AND gates and size of these AND gates is smaller compared to the previously implemented using ROM. So now I will have (a, b, c) as inputs and the AND array and get out of this the product terms P_0 P_1 P_2 P_3 P_4 and this I will call P_0, this I will call P_1 and this P_2 (Refer Slide Time: 16:20) and then my OR array would tie, there are three outputs, this is my f_1 f_2 f_3.

(Refer Slide Time: 16:30)



$f_1(a,b,c) = \Sigma m(0,2)$
$f_2(a,b,c) = \Sigma m(0,3,4)$
$f_3(a,b,c) = \Sigma m(3,4,5,7)$

a \ bc	00	01	11	10
0	1			1
1				

a \ bc	00	01	11	10
0	1		1	
1	1			

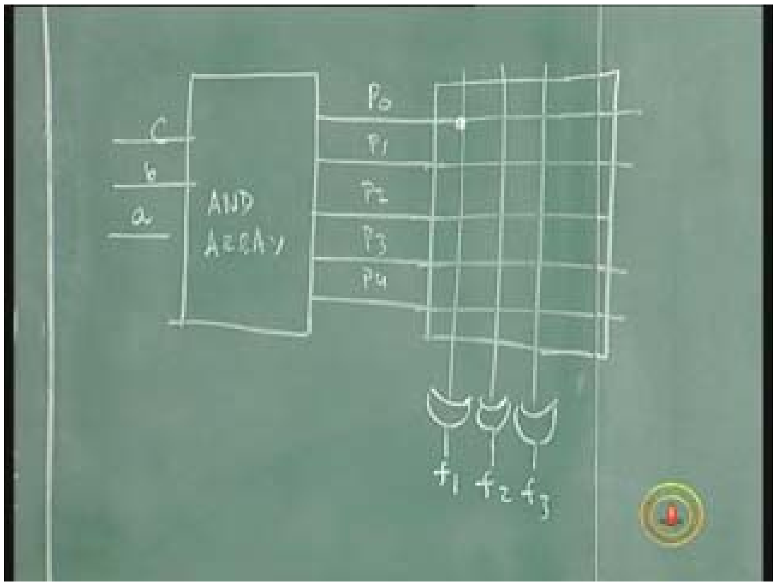
a \ bc	00	01	11	10
0			1	
1	1	1	1	

$f_1 = \underbrace{\bar{a}\bar{c}}_{P_0}$

$f_2 = \underbrace{\bar{b}\bar{c}}_{P_1} + \underbrace{\bar{a}bc}_{P_2}$

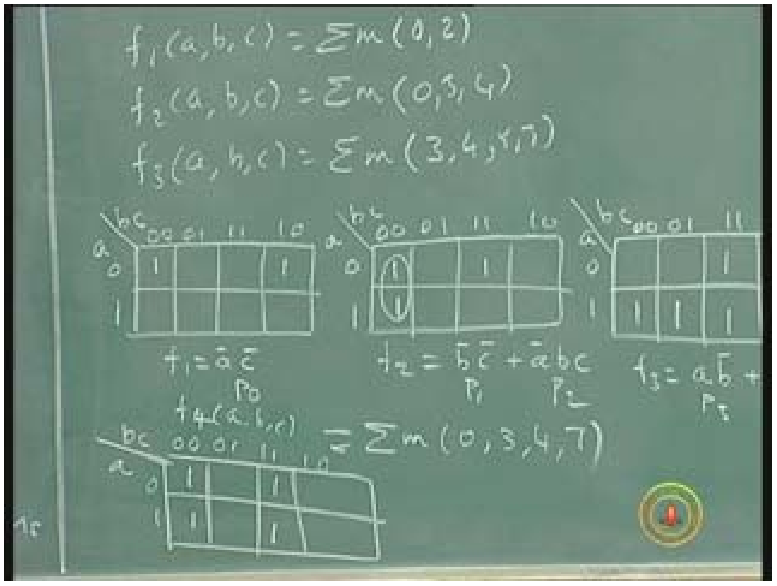
$f_3 = \underbrace{a\bar{b}}_{P_3} + \underbrace{bc}_{P_4}$

(Refer Slide Time: 17:23)



My f_1 will be P_0; f_2 would be P_1 P_2; f_3 would be P_3 P_4 so now I have the implementation of the function, the size is now smaller for the same function four functions. Had I used a ROM I would have used eight min terms and four outputs so the size of the ROM would have been 2 power 3 times 3, let us not say the whole number because we do not know how many rows and columns are there, so it is 2 power 3 times 3 so only then you will know you can take a ROM with three inputs and three outputs three address lines and three output terms. So size would have been 2 power 3 here it is not 2 power 3 but five product terms times three so you will have number of bits the size of bits is smaller here and the min term combinations from the AND array is also smaller now because it requires to do only five terms rather then eight terms with varying sizes and gates.

(Refer Slide Time: 20:00)

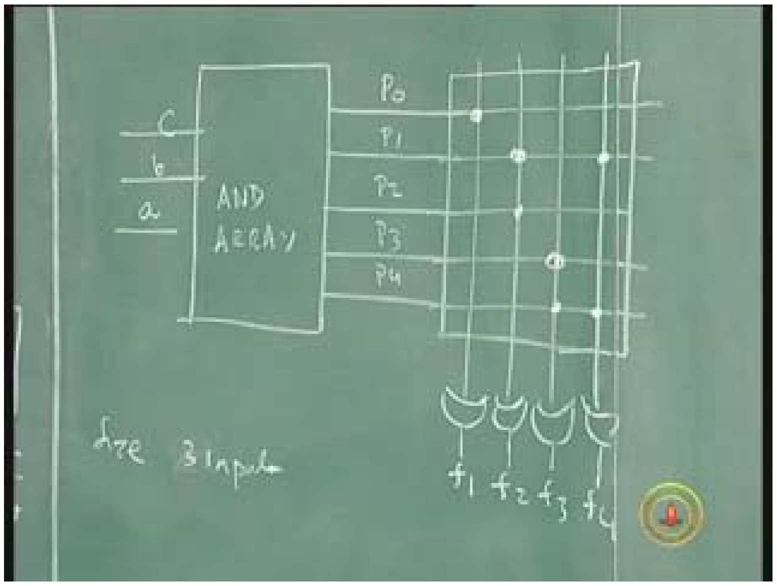

$f_1(a,b,c) = \Sigma m(0,2)$
$f_2(a,b,c) = \Sigma m(0,3,4)$
$f_3(a,b,c) = \Sigma m(3,4,5,7)$
$f_1 = \bar{a}\bar{c}$
P_0
$f_2 = \bar{b}\bar{c} + \bar{a}bc$
P_1 P_2
$f_3 = a\bar{b} +$
P_3
$f_4(a,b,c) = \Sigma m(0,3,4,7)$

Now if I add one more function to this f four suppose now I have a four output so I need to simplify this map also, this should be 0 1 2 3 4 5 6 7 now what is this? F_4 is b bar c bar or bc and b bar c bar is already there which is P_1 and bc is already there which is this P_4 segment which is same as the P_3 plus P_4. So with these product terms that are already generated four product terms already generated P_1 P_4 I am now able to get one more output and that is why I drew a larger box (Refer Slide Time: 21:01) so I can insert this, f_4 which would be P_1 P_4 this is what I mean by common products terms.

If you have common product terms between two functions you don't have to generate it, this term P_1 and P_4 need not be generated again because they are already there so all I have to do is use them. So we have the advantage because the AND array size is smaller and not only the number of gates but the size of those individual gates, the number of inputs to each of these gates will be significant to the number of inputs with practical systems.

Just because we take a simple example in the class and complete it in a short time draw any diagram on the board does not mean all systems are like this, these are only conceptual diagrams. In the real design you will many more input variables and the number of product terms, if you want do a ROM based design the increase exponentially explodes so there it is a real advantage because however big the system may be namely the number of inputs the number of outputs a given output will be dependent on a few inputs but not more than that a few inputs at a given time, it will be a localized input for that particular output then another output will move and then one more set up. So why generate everything every time and only use of few each time, that was the concept of programmable array logic, the size of this would be, here (Refer Slide Time: 22:49) you have to say the size is four products terms I should have the capacity to have three inputs,

so how do you express the capacity here? You express the capacity as 2 power n times m where n is the number of inputs and m is the number of outputs.
(Refer Slide Time: 23:04)



Therefore, if you want to do a ROM based design then don't do a multiplication of 2 power n times then you do not know. You cannot go to a shop and ask for a 2 power 3 times 4 and 8 times 4 is 32 so you cannot go and say you give me a 32-bit ROM. We can say give me an 8/3 ROM or 3 input by 3 output ROM. Like that in this case number of product terms it must make and the number of outputs. So I should give here three inputs five products terms I should be able to generate five product terms using these three input then I need four outputs.

Student Professor conversation (Refer Slide Time: 23:55 – 24:30)

Yeah, I am just giving you an example of how a function can be shared that is why I have these three which are different, I cannot add to anything any time even now, adding is not after you design, this was a requirement and now this is another requirement I am showing you the same IC can also do the job provided you have one more output.

Student Professor conversation (Refer Slide Time: 24:30 – 24:50)

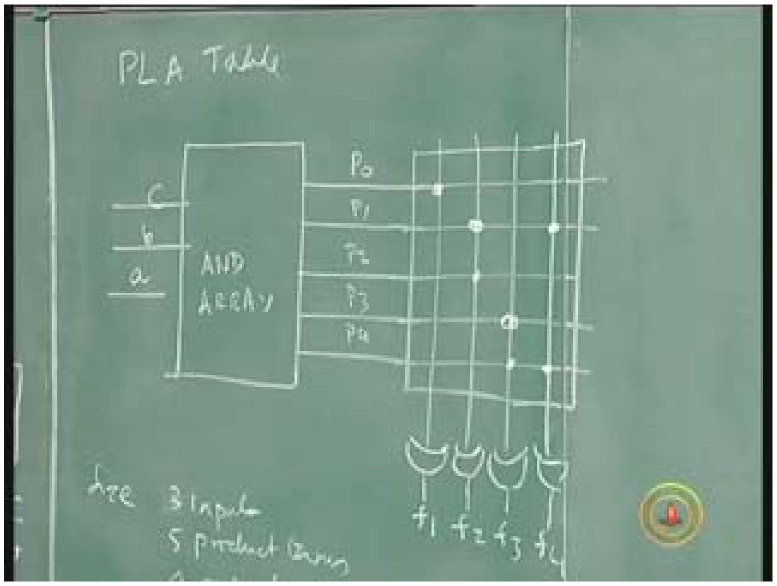
The manufacturer will give you general sizes, manufacturer will give you three input, five product terms and four output schemes, of course you cannot have all possible inputs and all possible product terms and all possible output terms, infinite number of combination he will not give you. He will give you a reasonable choice.

I have a four term PLAs Programmable Logic Arrays available, 6 are available, 8 are available, 12 are available, 16 and so on so you choose. Similarly they will say for this 6 I have a variety of product terms 4 8 6 10 and for each one of them they will also give me four out put so the hole variety of output, commercial products available so you have choose the best one. it is not that you go to the shop and get a particular device that you have in mind, I will put it the other way, I should have a Programmable Logic Array which will at least have three inputs and can at least generate five product terms and at least have four outputs and the smallest within that constraint I will buy.

Supposing there is no three input Programmable Logic Array PLA is available I will use a four input so supposing a four input is available I will take that and use three of those inputs otherwise how will I do this job. Even for ROM it is the same thing, I cannot say seven input ROM, two power seven size may not be available, standard inputs 4, 8, size of the ROM is not available at your will there are fixed sizes you will have to choose from them, if you cannot fit it with this smaller ROM then go for a bigger ROM. It is not like the notebook that you want to buy from a store, you want a 100 pages or 200 pages, can you buy a 147 pages notebook because the teacher is going to teach 147 pages it is not possible, when you want to buy Coco Cola also you have to take a 6 pack or a 4 pack and you cannot have a 5 pack, it is always the case.

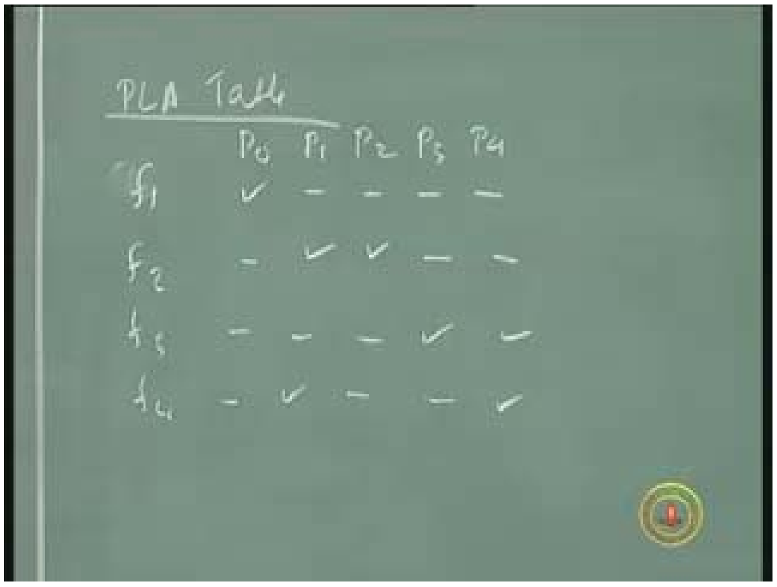
If you are doing the design you have to go to the shop, the problem is given to you, I have a problem with three inputs, four outputs following the truth table, I sit down take a piece of paper and pencil, reduce these product terms and then come up with a list of product terms for each output then I come with this table PLA table; input a b c product terms to be generated and product terms to be combined for f_1 f_2 f_3 they are called the PLA table Programmable Logic Array table, you have to generate a PLA table which will give you inputs, the products terms, suppose I can say the PLA table will be like this in this case, I will write PLA table for these functions.

(Refer Slide Time: 27:19)



The PLA table would be P_0 f_1 f_2 f_3 f_4 and P_0 P_1 P_2 P_3 P_4 so f_1 will only have P_0 and f_2 would be P_1 and P_2 and f_3 is going to be P_3 P_4 and f_4 would be P_1 and P_4. Therefore for the given system the inputs are given.

(Refer Slide Time: 28:20)



PLA Table

	P_0	P_1	P_2	P_3	P_4
f_1	✓	–	–	–	–
f_2	–	✓	✓	–	–
f_3	–	–	–	✓	✓
f_4	–	✓	–	–	✓

I will also note P_0 P_1 P_3 P_4 are functions of the three variables. Even though you may not use all the three variables they are functions of the three variables (a, b, c). If this is the table I have got by designing it then this is the information from where I start I know that

I need three inputs, I know that I need five product terms and I know that I need four outputs so I will go to the electronic component store and ask for a PLA with three inputs, five product terms and four outputs, I have four inputs, six product terms and four outputs, so I want to take it if I want to really do the design but we really don't do that, actually there is a catalog available, it used to be a big thick book when we were students but now it is all in web so if you go to the web and say this is my requirement so they will spit out the part number of different manufacturers and then you choose that person and that particular device.

If this is the minimum requirement then give all the possible combinations then they will give you like there is one like this, four inputs, six product terms and four outputs would you like to take it then you say yes. Probably today you can order it online and give your credit card number and only thing they have not invented is to deliver the IC through the computer but all other things are there, somebody has to deliver it through courier, but that system will probably come out sometime later, you open your computer then you pick the IC you want, probably that is left for you people to research on when you come to it.

(Student Professor conversation 30: 40) everything is programmable, product terms are programmable and outputs are programmable, what is programmable is which of the product terms you combine is programmable I can have up to five terms in each of my outputs, any combination. I can have f_1 is equal to $P_0 + P_1 + P_2 + P_3 + P_4$ again $f_3 = P_1 + P_0$ of course you won't do it you do not want all the four outputs to be same, in principle I can have each of them outputs and have 5 products terms but my specification requires only these types of things.

Up to five inputs I can connect and each of the product terms go as an input to the OR gate for each of the functions and there is the programmability similar to the ROM programmability. the ROM programmability you understood in the last lecture it is the same and there is no change but the additional programmability instead of getting programmability m zero m one m two m three m four m five m six m seven and m sixteen if it is four inputs I am now going to have P_0 P_1 P_2 P_3 stop because I need only four product terms sharp P_0 P_1 P_2 P_3 P_4 P_5 stop so how many products term we need in order with taking it account to common to possible so I will do a simple in fact we talk about in the design of combinational logic also, when the multiple output combinational logic is designed for the same set of inputs you try to share the terms to the extent possible. Sharing the terms will give you reduced hardware.

Supposing I having four inputs and seven outputs in the case of binary to seven segment encoder there if I can share some of those terms for each of those outputs then to that extent I do not have to reproduce that gate again, same concept applies here.

Even if you leave the sharing part out I don't have to generate seven product terms I only generate as many product terms as required for my implementation. So, to that extent there is a saving in the number of AND gates and the number of inputs to each of those AND gates. Earlier in the ROM case I needed to have eight of those AND gates and each of those AND gates had to have three inputs. So when you take a two input case or three input case it is very hard for me to justify, looks trivial and marginal but as I said many practical systems we have huge number of inputs and you want sixteen input combinational logic then what will be the size of the ROM? Number of words is 2 power 16 = 64k. Now if I can produce the whole thing in fifteen product terms what a difference from 64k to 50 the dramatic change in the hardware required. This is the reason it is called programmable and not in any other sense.

I choose my product terms instead of blindly taking all the min terms and here again I choose my product terms to go into each function, in that way it is programmable and un-programmable. now there is one more product of the same type Programmable Logic Device which is called for some reason, they can’t think sometimes, they didn’t know how to name it, first they called Programmable Read Only Memory because all the possible inputs can be given and all the possible outputs then they said logic array for example then instead of changing this programmable logic array they interchanged the two words, the last, the third variety is called programmable array logic PAL, the other one is PLA and this is PAL.

What is the need for a third variety? Here in the ROM I have one programming feature. once I know the size of the design the number of inputs and the number of outputs I go ask for a ROM of 2 power 7 times 8 or 2 power n times m get it and I know all the min terms are available for each of those m min outputs. It was easy to do it even though your hardware is inefficient; the efficiency of the hardware is not good. So they thought we will save on the hardware and they said I will do a two step programming, first I choose what product terms I need and then choose it and then I will combine them so it is a two step programming. again when the size is very large, number of inputs and the number of outputs are large, number of product of terms are large like 50 inputs and then 50 outputs first of all you have to reduce them properly and then we have to find if possible as many common terms is possible.

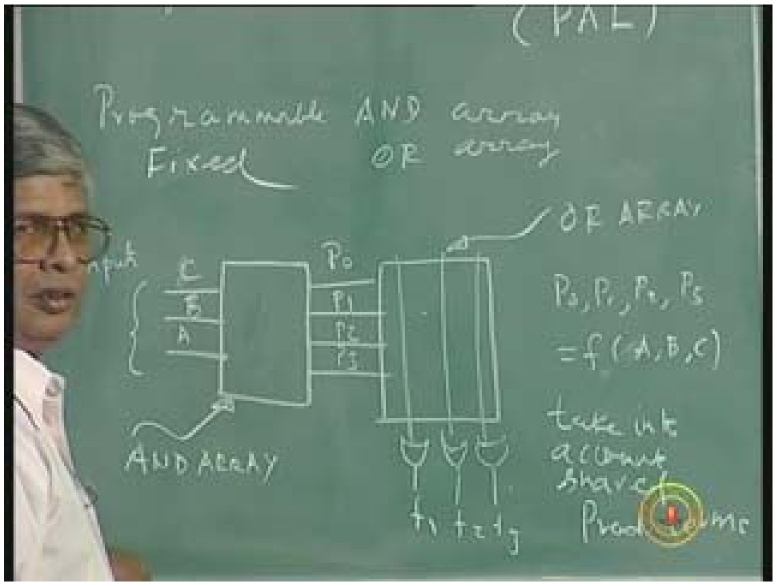
Of course everything is programmed nowadays in computer, the other programming, computer programming. Instead of manual labor anything which is of repetitive nature, checking for example computer does it better than us because it doesn’t get tired of it, we get tired and make mistakes, the number of things we do is very large. But anything in which you need to apply your mind you are better than the computer hopefully. Even doing that also people were very inefficient, I need to balance this and balance that and I may always not end up with an efficient design so ended up with an inefficient design.

Some of the Programmable Array Logic, it is all market driven, one should remember electronics or any industry for that matter pharmaceutical industry is market driven, it is a low cost, even though everybody else wants it but it should be affordable and people would buy it, so it is market driven. Even though the ROM solution was costly quote and quote costly those days but these days it is not costly but it is more area, why do you want to spend more area in power, they thought this solution will catch on but it was a failure, the market part was a failure, lot of people had difficulty in using this, they had to generate software to do this always, whatever I am saying will be done, reduce it find out the common product terms all that I am saying but in a large system all of those things will be computerized automatically, so they found it difficult to write efficient software for that so there were not many takers so they thought that is like giving too much freedom.

Earlier we had restriction, the programmability was restricted and the only difficulty was we had too many things and we didn't want so many min terms because many of them were not used. So they gave another degree of freedom two degree of the freedom is given, first degree of freedom, the AND gate selection programmable second degree freedom or the OR gate selection.

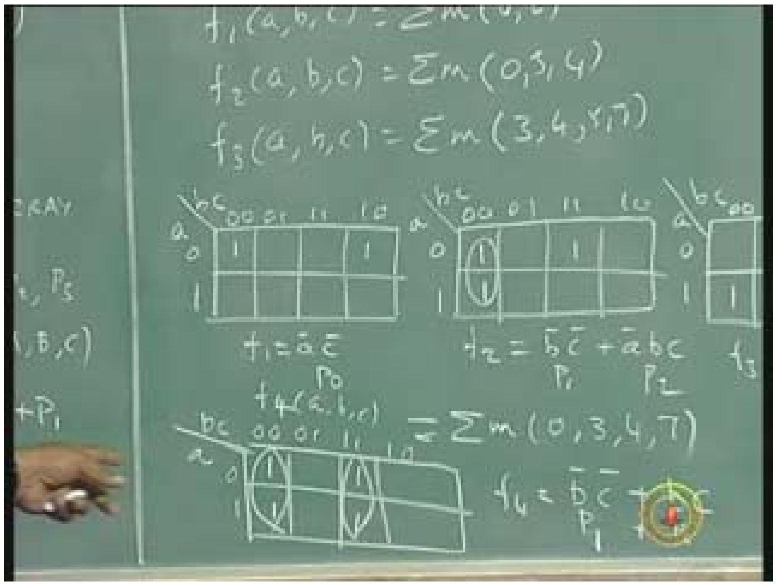
But you know as it happens in our country too much freedom also spoil the whole thing. So they found it difficult to have people work with this comfort so they put a restriction now again back to square one that I will not give you two degrees of freedom I will give one degree of freedom but the ROM accesses was not good because we had wastage of hardware, this is an inefficiency in hardware, ROM was in use so I will combine these two by taking the one degree of freedom from this so leave the AND array combination to you and the OR array gets fixed so that is the programmable AND array and fixed OR array.

(Refer Slide Time: 39:00)



So now in a ROM connection again I will do same thing, I will do P_0 P_1 P_2 P_3 as functions of f (A, B, C) I will generate the product terms required but what will happen is their prefix it will say f_1 will always be P_0 P_1 you have no choice on that. So you have to somehow get your P_0 P_1 f_1 reduce to P_0 P_1 and if it not possible you cannot use it so you go for a bigger one and f_2 may be P_2 P_3. Since I want to use the same drawing I will make it smaller (Refer Slide Time: 40:16), I fix which AND terms which product terms go into each of the OR gates so that is fixed it is not negotiable here, the product terms you can combine in order to make an output but want you can negotiate is the combination of those product terms itself the generation of those product terms itself, it is not a min term it is a product term. This is the product term and min term. the min term will have all the variables a b c d e f along with its compliments whereas the product term will only have those letters that are required for that particular inputs.

(Refer Slide Time: 40:57)

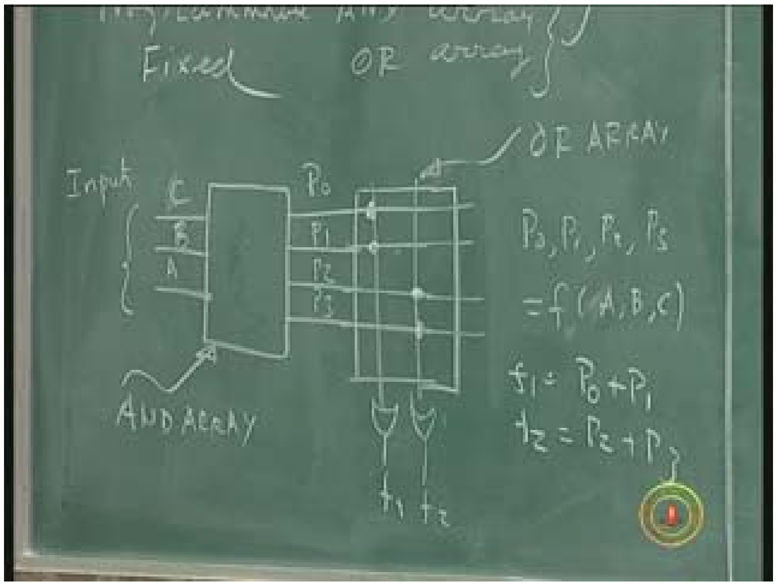


$f_2(a,b,c) = \Sigma m(0,3,4)$
$f_3(a,b,c) = \Sigma m(3,4,5,7)$
$f_1 = \bar{a}\bar{c}$
P_0
$f_2 = \bar{b}\bar{c} + \bar{a}bc$
P_1 P_2
$f_4(a,b,c) = \Sigma m(0,3,4,7)$
$f_4 = \bar{b}\bar{c}$

Therefore, now there is no question of the common product terms. that is why I removed this common product term restriction (Refer Slide Time: 41:06) so I do not have to look for whether something is common between this and this because even it is common I have no way of connecting this and this.

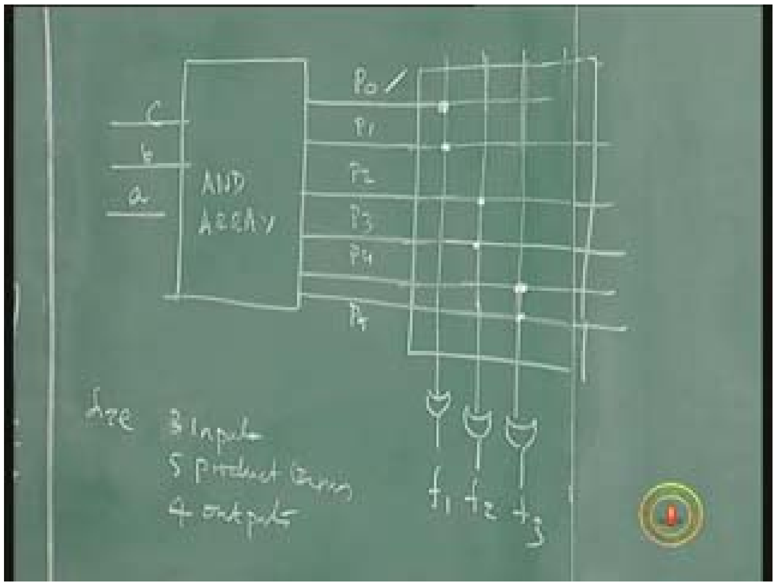
Supposing P_1 was common to f_1 f_2 I cannot combine it because P_1 only is fixed to f_1 and P_2 is fixed to f_2 so I have to make P_1 P_2 same that means I have to regenerate it. I have to generate P_1 once for this and once for this.

(Refer Slide Time: 41:35)



I might call it P_2 for convenience but I need to say that if the same AND term same product term appears in two outputs I need to generate it for each of those outputs so there is no question of sharing so that takes the burden off, when the burden is removed from you then you are happy, at what cost a little less efficient that's okay, hardware is not a big deal as we said today. So this is what it is, the AND array is programmable, the OR array is fixed so I will use the same example etc I am going to removing my f_4 because f four i just wanted to show you how to exploit the common terms, there is no question of exploitation here because if I want to produce no P_0 P_1 P_2 P_3 P_4 I need to generate to five product terms call P_0 to P_4, I have to combine f_1 as P_0 this is what it is going to be (Refer Slide Time: 43:00) so for each of these product terms each of these output has two product terms as inputs, I am going to have an extra product term called P_5, I will re-designate this because f_1 can be only P_0 or P_1, there is only one term required for P_0 I will not use the other term, the other inputs will not be used.

(Refer Slide Time: 43:50)

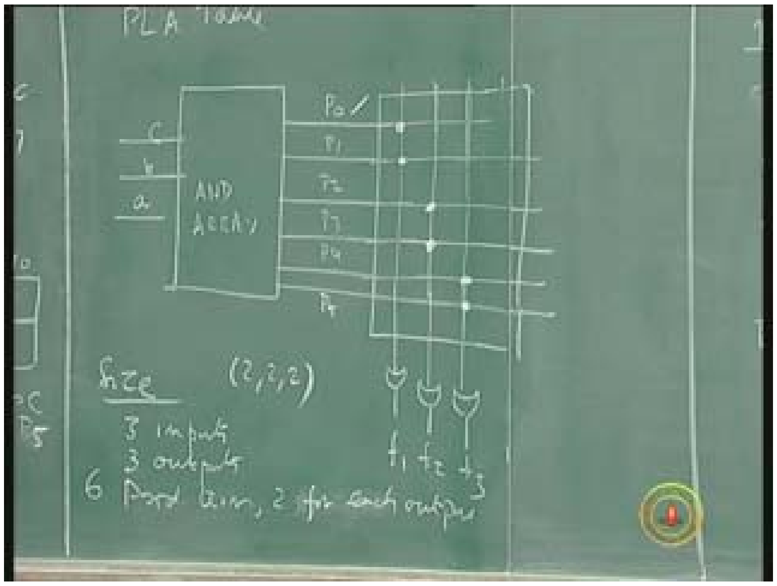


So P_0 will be used, I am not going to combine anything, I don't combine anything because AND is programmable by S, when I program the AND I will only program for P_0, I will not program for the second term of this. so I am going re-designate it as P_2 and P_3 because they belong to the second output, I am going to re–designate it as P_4 and P_5 so these two will be f_1 f_2 f_3 so P_0 plus 0 so the second term here is not being used. I have only one term requirement but I have a two input gate I can't do anything about it, I will not use the second input, here I will use the two inputs.

Now if I want f_4 which was originally thought as b bar c bar or bc in which case b bar c bar is P_2 but I cannot use this P_2 here. I have to re-designate it to P_6 and P_7 and again generate P_6 and seven even though P_2 and P_5 is available I cannot use them because I don't have this input tied to next. So, if I have a fourth OR gate with two inputs those have to be again programmed for the same variables which are already there then it becomes less efficient to be on that.

So how will you define the size of this? The size of this should be defined a little more carefully. Size should be three inputs or three, there is no content to these inputs and outputs, but I need two OR gates as inputs to each of these, two AND products so I need to have total of six product terms two for each output. As there is a standard way of writing it they will say 2 2 2 product terms so when they write 2 2 2 product terms I have three outputs first output will have two product terms, second output will have two product terms and so on. Again I have 2 3 3 or 2 3 3 2 where I have four outputs, first can be 2 inputs, second can be 3, third can be 3, fourth can be 2.

(Refer Slide Time: 46:40)



So again you go through the manual if you want to do it the manually or you want to do it in computer, give the requirement, here it is even simpler, I don't have to worry about the commonality of these product terms. I am giving the function f_1 f_2 f_3 and f_4 all functions are given, I am told they are four variable functions or three variable functions or five variable functions whatever it is, I make simplification using Karnaugh Map and then list them all out and suppose I end up saying for f_1 I need three product terms, f_2 I need four product terms, for f_3 I need 3, f_4 is 2 then I say imminent function with three inputs of four inputs four outputs one two inputs, two three inputs and one four inputs, it may not be their exact match then you go and get the nearest possible size and do not use like in this case I have two inputs (Refer Slide Time: 47:58) out of which I am not able to use P_1 because my f_1 requires only P_0 but P_1 is not used, like that I may have another output OR gate with three product terms I will may have only two so I have to put one f_1 and use.

So the point is the design becomes simpler because there is only one programming step namely in the AND gate choice the product term selection, OR gate gets fixed and the second burden of having to identify some common terms becomes less so this is got up very well. Hence, Programmable Array Logic is a standard PLD design today. so this in an evolution I talked about today, the PROM design, that is PLA design then PIL design, historically it evolved in this way but what is now popular is if you want to do a combinational design using Programmable Logic Devices the most preferred Programmable Logic Device for combinational design today is Programmable Array Logic or PAL.

As I said there are some problems but then they say it is simple, you can choose the one you want exactly, it is also efficient, earlier we had a variety, sometimes in a variety you have problems you make an inefficient choice but here an efficient choice is possible.

So what you have to see a little more into this is, the most popular Programmable Logic Device namely PAL Program Array Logic, if you look at this a little more closely and see what are the various types of PILs available and how to specify what we need there standard procedures for specifying what we need and so on. We need one more lecture on this to complete the Programmable Logic Devices. We will see it in the next lecture.