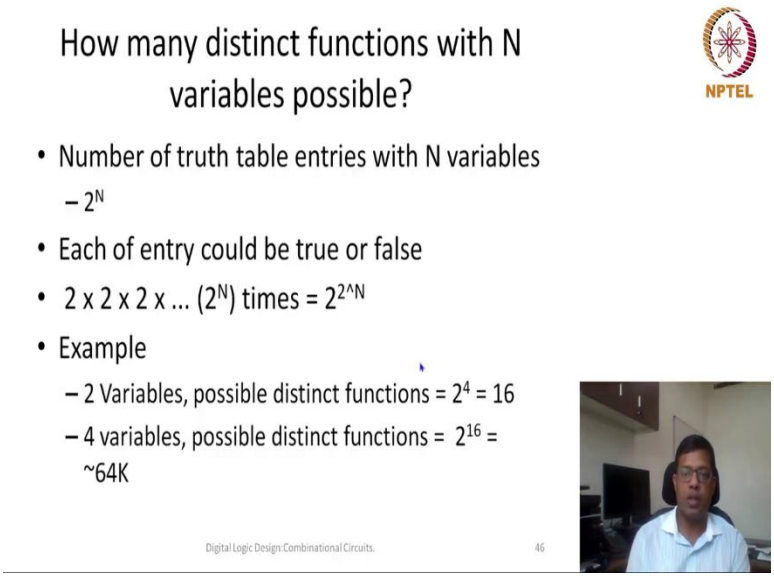
Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Programmable Hardware

Hello everyone, today we are going to discuss about a programmable logic that can be designed using combination logic. In other words we would like to learn how to design a generic logic which can be used to design any logic. Programmable means that we can program it, once chip has been fabricated. So, once VLSI chip or application specific chip has been fabricated then you cannot do any change in the logic, but what we would like to use that particular chip we would like to mimic that any logic using that particular chip.

To understand this the gravity of this question how can we design any logic which is programmable or any logic which any circuit which can behave like any other logic. So, to understand this gravity let us ask one question from our self that if we want to create an n variable function, how many distinct possibilities are there? Shall I repeat this question that I want to design an n variable function, so how many such functions how many different distinct such functions are possible?

(Refer Slide Time: 1:49)



So, to understand this one thing which we can start with this that if there is a function with N variables then certainly we can design all of these functions using truth table. So, my truth table

will have 2 to the power N entries. So this is my starting point of my thinking, so that if a function I want to design it will have 2 power N entries. Now, what is a function? So we have understood that of a boolean function in a standard form can be represented as some of mean terms or product of max terms.

So, let us consider the first ones some of mean terms. Now, each entry in this truth table represent one mean term. In a function all of these mean terms N or a particular mean term could be present or could be absent. So from that perspective then each of this mean term could be present or absent in my function or its output could be true or false.

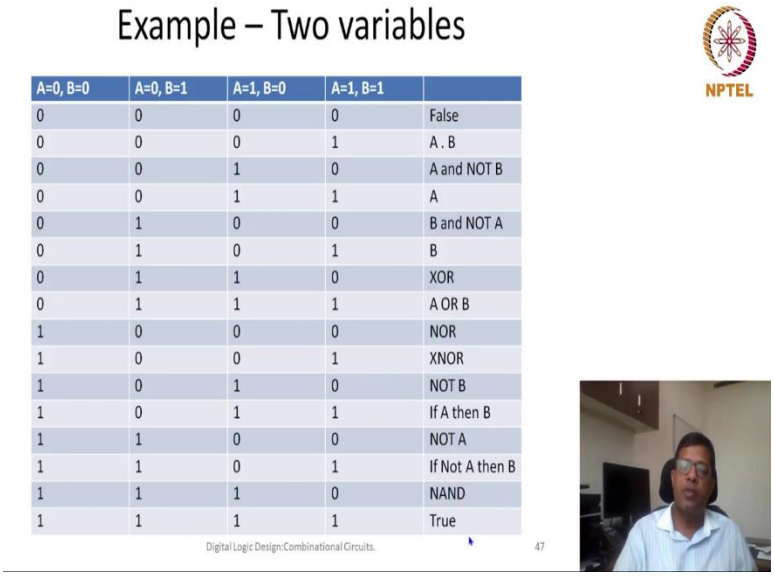
So in that way the total number of possibilities would be 2 into 2 into 2 up to 2 is to power N. So, I can write it like this 2 into 2 into 2 up to 2 is to power N, so essentially the total number of such possibilities would be 2 into power of 2 into power of N. So how big that number is?

Let us say, if number of variables are 2 distinct possibilities are 2 is to power of 4 only 16. Now, if the number of variables are 4 then the different possibilities of different functions being possible is 2 is to power 16, from where 16 came? Because 2 is to power 4 is 16, so total number of truth table entries in a 4 variable function would be 16 and because all of these entries could be present or absent so total number of functions which are possible is equal to 2 is to power 16. Which is approximately equal to 64000.

So, if I want to design a function of only 4 variables and I want to have either have all the 64000 possibilities so that I can pick and choose that this is the function I would like to design or I have to design something which is generic. So that I can say that this particular generic unit can implement any function, any of these 64000 functions of 4 variables which has 4 inputs any of these 64000 functions can be implemented but yes only 1 at a time.

This is what are programmable logic would be and this is what we will see in this lecture. Now, but before this my mind is still not clear that how is it possible that there would be 16 such distinct functions just with the 2 variables.

(Refer Slide Time: 5:22)



A=0, B=0	A=0, B=1	A=1, B=0	A=1, B=1	
0	0	0	0	False
0	0	0	1	A . B
0	0	1	0	A and NOT B
0	0	1	1	A
0	1	0	0	B and NOT A
0	1	0	1	B
0	1	1	0	XOR
0	1	1	1	A OR B
1	0	0	0	NOR
1	0	0	1	XNOR
1	0	1	0	NOT B
1	0	1	1	If A then B
1	1	0	0	NOT A
1	1	0	1	If Not A then B
1	1	1	0	NAND
1	1	1	1	True

So, let us see in detail with the example that yes is it actually possible or not? So now what I do is I will try to write all possible truth tables, so how do I write all the possible truth tables? So essentially how many entries would be there if the number of variables are 2? Truth table entries would be 4. So, 1 truth table entry would be corresponding to A equal to 0, B equal to 0 other entry would be corresponding to A equal to 0 B equal to 1 and similarly A equal to 1, B equal to 0 and A equal to 1, B equal to 1.

So there are the 4 entries in my truth table. Now what I am try to do, I am trying to enumerate all the possibilities with these 4 outputs. So essentially each row in this table correspond to a different function. So this is one function where my output, my function f has all the 0 all the outputs are corresponding to all the truth table entries all the outputs are 0.

Similarly, this is the other combination where, when A is equal to 0 B equal to 0 output is 0, when A equal to 0 B equal to 1 output is 1, when A equal to 1 B equal to 0 output is 0, A equal to 1 B equal 1 output is 1. So, if you see this table carefully then it is nothing more than counting from 0 to 16, so this is 0 0 0 0, 0 0 0 1, 0 0 1 0, 0 0 1 1 so on so forth.

So, I have enumerated all the possibilities with these 4 outputs. Now, the second question also came immediately, will all these function we have seen that there are 16 possibilities, will these

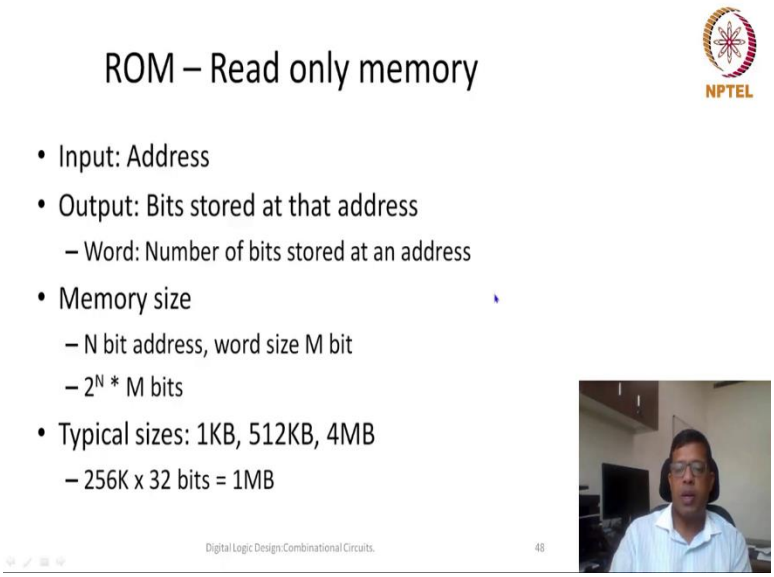
16 possibilities make sense? Are they actually distinct? Interestingly for this 2 variable case they are not only distinct but they also make different logical sense.

So, for example all the output my truth table output for all these combinations are 0 so that means I can say this function is false, f is always going to be 0. Similarly, if first 3 are 0 and it is only A equal to 1 B equal to 1 I can say it is actually a truth table for AND gate. Now, similarly for this particular combination I can say it is A and, and it with not of B.

So, if both of these two are A, these twos are 1 and these twos are 0 so that means it actually represent it is A. So, similarly you see all of these B and not of A or B XOR A or B NOR XNOR not of B, so all of these are meaningful combinations. So, these are the different 16 combinations which we can create using 2 variables if the number of variables are more than the number of combinations will also be more.

So, for example as we said in our previous slide, if number of variables are 4 then 64000 little more than 64000 combinations or functions distinct functions can be created based on what the truth table is. So, this is the premises of this lecture, so now let us take a small detour and we will first understand what is a random sorry ROM Read only memory.

(Refer Slide Time: 8:59)



So, although we have seen in one of the previous lecture we have done some of the exercise, design exercise using memory so we are trying to repeat here but the question here or the topic which we are trying to understand is Read only memory. So that memory which cannot be

written but can only be read. So such a memory will give input as a address so there would be all those contents at those addresses and when address is given then bit stored at that particular address would be written.

So, since I am using word bit so that means there is a possibility that more than 1 bits are stored at that particular address. So, we call the number of bits that are stored at single address is called word. Word could be combination of 1 bit 2 bit 4 bit 16 bit 32 bit 64 bit. You can also ask that why I am only saying the number power of 2. Actually word could be 24 bits also, it could be 23 bits also but usually it is power of 2 because in the binary word it helps us in lot of computation.

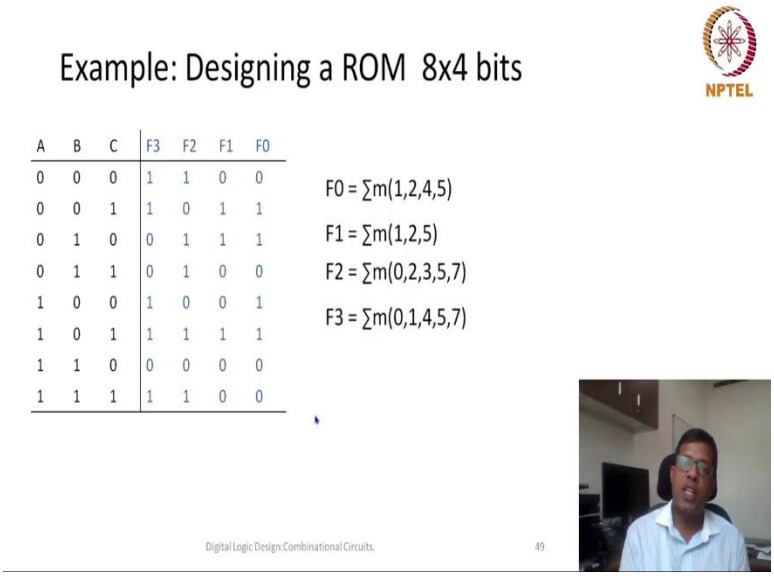
It helps us in alignment of addresses bits, so because competition becomes easier when it is power of 2, so that is why most of the time word or the number of bits which are stored at the particular address is power of 2. But not absolutely correct so it could be (())(10:49) power of 2 also in some specific cases.

So what is inside ROM? That is what we call the total number of memory cells, total number of content. So, if N is the number of bits in my address and word size is M bit then I can say that the total, content total number of memory cells is going to be 2 is to the power N into M bits. At every address how many bits are stored? M bits and how many such addresses are there?

If the address bit number of bits in the address is N the total number of addresses would be 2 is to power N. So, 2 is to power N into M would be the total memory cells which are present in this read only memory. Now, what could be sizes? Usually when you see memory, you see it is written as 1 kilo byte, 2 kilo byte it is again usually power of 2, 1 kilo byte, 2 kilo byte, 512 kilo byte, 4 Megabyte, gigabytes, terabytes, petabytes, Hexa bytes.

So, but when we are writing here we are saying 2 is to power N into M so where is this M? So, M means it is usually contained inside this byte information. So, for example there could be 256 kilo addresses into 32 bits then it will form 1 megabyte. So, similarly it could be the other way around also, so there could be 1 M addresses and each word says is 4 bits, then the size would be 512 kilo bytes.

(Refer Slide Time: 12:48)



So, given this background on read only memory then I want to design it. Now, let us say I want to design something like 8 into 4 bit memory so the total number of locations are 8 and on each location 4 bits are stored. Now, such a memory can be represented using a sort of a truth table. While putting this truth table actually this part which is written in black is redundant not required, we can simply have what is written in blue.

So, each line can represent 1 memory location and the subsequent memory locations could be subsequent contained at the subsequent location can be written in every subsequent line. So, the assumption there would be that the first line means addresses 0, second line mean the address 1, the third line mean address 2, so on so forth.

So, only this content would be sufficient to show what is there inside ROM. You are saying because it is a read only memory content is fixed, nobody is going to change the content it cannot be changed because content cannot be changed so we can write the content in sort of a file or a basically an array and we say that this is the content on my read only memory.

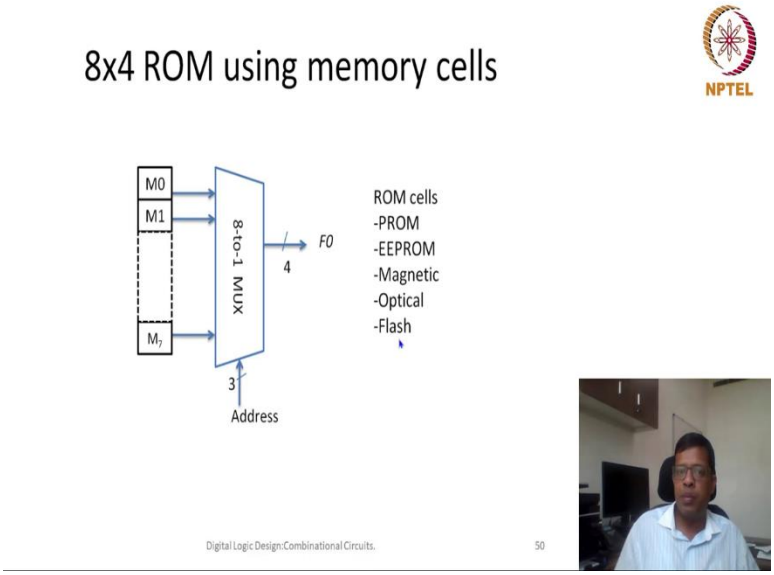
Now, if we specify the content at each and every address then it looks like a sort of a truth table, now I want to design such a read only memory. How do I design it? It looks like a truth table, so because it looks like a truth table so what I can do is I can create 4 different functions. So I can that my F0 is actually the sum of mean term 1, mean term 2, 4 and 5.

Similarly, function F1 is sum of mean term 1, 2 and 5. My function F2 is sum of mean term 0, 2, 3, 5 and 7 and function F4 is a sum of mean term 0, 1, 4, 5 and 7. So after writing this sum of mean term expressions then I can use k map to minimize it and find out what are the functions for corresponding this F0, F1, F2 and I can write them in form of A B and C.

So this way I can design a ROM. So, in other words what we are saying that for designing a ROM, designing a read only memory I can use my combinational circuits. I can use my AND and OR gates to say that yes this is actually a ROM. In other words we are also saying that a ROM, your memory cells could be defined by your combinational logic. There is no memory cell it is a perceived memory cell.

If we are saying 0 0 0 my output is 1 1 0 0 so we are perceiving that this 1 1 0 0 is stored in some sort of memory cell, but actually it is coming out of a, my combinational circuit or a circuit which is made of AND OR NOT etcetera, basic gates.

(Refer Slide time: 16:46)



But yes it is not always the case that we always design ROM using this basic circuit there is also a possibility that we can use memory cells and if we are using memory cells then we can design it something like this that let us say we have this memory cells, we have the content of all from 0 1 2 3 7, 0 1 2 3 4 5 6 7 so whatever is the content that is stored in these memory cells and we can then access it using a multiplexer.

So, output of each would be given to multiplexer and the 3 bit address A B C will go here and output would be F0. Similarly, we can have the memory cells for all other F1 F2 F3 F4 and total number of memory cells would be 32 and corresponding memory cell will go to 4 different multiplexers and F0 F1 F2 and F3, 4 outputs will come. So there also are very commonly used design of a ROM using a memory cell.

Now, if this is how we design ROM then what would be the memory cell looks like? Now, there various possibilities for these memory cells. One of it is called programmable read only memory, programmable means that whatever whether 0 need to be written or 1 need to be written so this 0 or 1 is written at a by passing some let us say a very high current or it may be a fuse.

Which says that, if it is short circuited then it is 1 if it is open circuited then it is 0 and using that short circuit and open circuit analogy so there could be like a high current will pass and then if material is of low resistance it will burn out so fuse will go away, so it will become short circuit sorry open circuit and otherwise it will remain short circuited and in usual operation the current rating would be less so it is programmable.

But once programmable you cannot change it, that is anyway the definition of ROM also, that read only memory you cannot change the content, but one thing that if content cannot be changed it is highly unusable. So, if we cannot change the content of read only memory you can never ever change the content of read only memory and you can only change at the time of designing or at the time of fabrication. Then what is the use of such ROM?

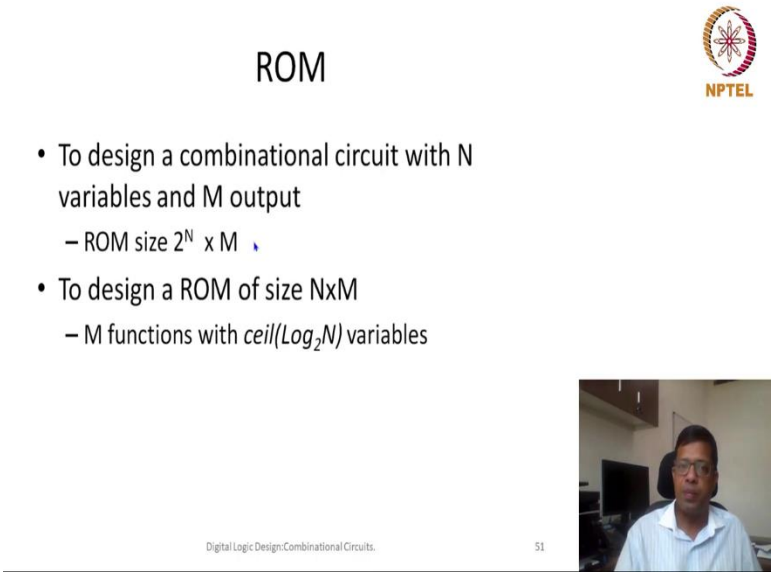
The use would be there but it would be very very severely limited. So that is why in last 30, 40 years we have doubled or we have thought about various techniques so that it becomes programmable and it is easy to program. So, one of the other technique which is used is called electrically erasable programmable read only memory.

So that means using some high current or high voltage we can erase it also and then we can program it again using some high voltage or high current. So, that means every time you would like to use that ROM then you can program it. The other methods whatever storage types you have, you have magnetic storage, the hard disk are usually magnetic storage or your DVDs or CDs they are all optical storages.

They are also sort of ROM that once you have written in your optical drive that is why it is also called CD ROM compact disk read only memory. You cannot write it again and again, so but once you have written then you can read it. Although now we have techniques to and these techniques are cheaper that you have this laser pointers which will be able to write to this optical methods also and we can read also. But usually reading is much faster than writing, writing you do once in a while and also all the ROM cells which are writable total number of writes are also very very limited.

So, let us say each location you can write only let us say 10 number of times or 1000 number of times or 100 1000 number of times. The last option which is given here this flash so you have this huge mistakes, so these are called NAND flash or NOR flash. So, all of them are also sort of read only memory and you can write them but once I can read them multiple times but you can write only some finite number of times. The value will remain there.

(Refer Slide Time: 21:45)



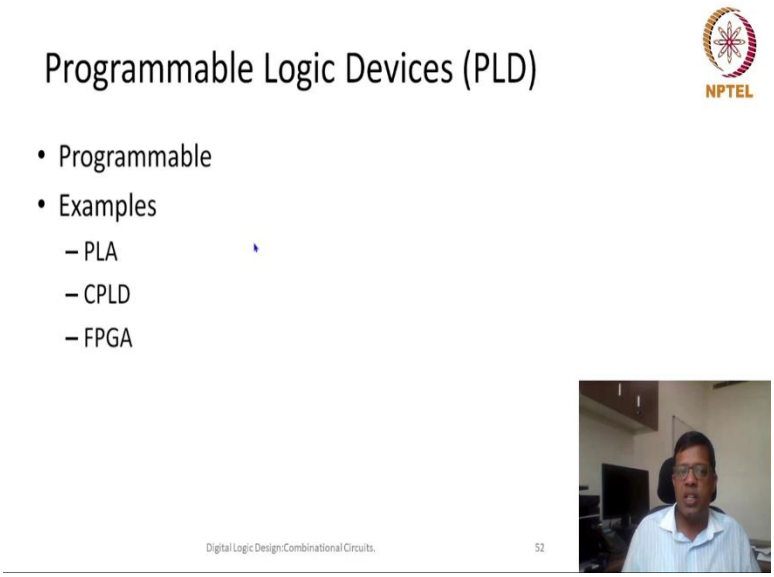
So, now these read only memories you can see that I can use it to design any combinational logic with N variables. So, we see that I can design a ROM by creating M functions of log and variables. I want to design a read only memory of size N cross M then I would require M functions and each of these M functions would have the variables which is equal to log N and on the other side also if I have a read only memory which I want to use to mimic a combinational circuit which is also possible.

So, let us say I want to design a combinational circuit with N variables and M outputs, so I can do it. I can design a combinational circuit without using gates I can design it using a ROM. The size of that ROM would be 2 is to power N into M. So, these many number of entries in the truth table and because each entry would have this M bits, so M functions would be possible.

So, this gave us a little more idea that if I want to create a generic method the original question where we started with this lecture that can we design a circuit which is generic enough and which can use to design any function of let us say 4 variable. So, then what we can do is we can use a ROM, we can say that this particular ROM will have 2 is to power 4, 16 entries and because this function has only single output so 16 cross 1 that many number of entries are there then we can implement or we can say that this particular function would work like this.

So, in other words this ROM can be used to design any generic combinational circuit with N variables M outputs. So, this is one method which can be used to design programmable logic. Further we can use EEPROMS electrical erasable so that we can re program them by passing high voltage, high current and some controllers. We can use flashes we can use other methods to have these read only memory and which is programmable as well. So, this is one method of designing programmable chips or programmable hardware.

(Refer Slide Time: 24:48)

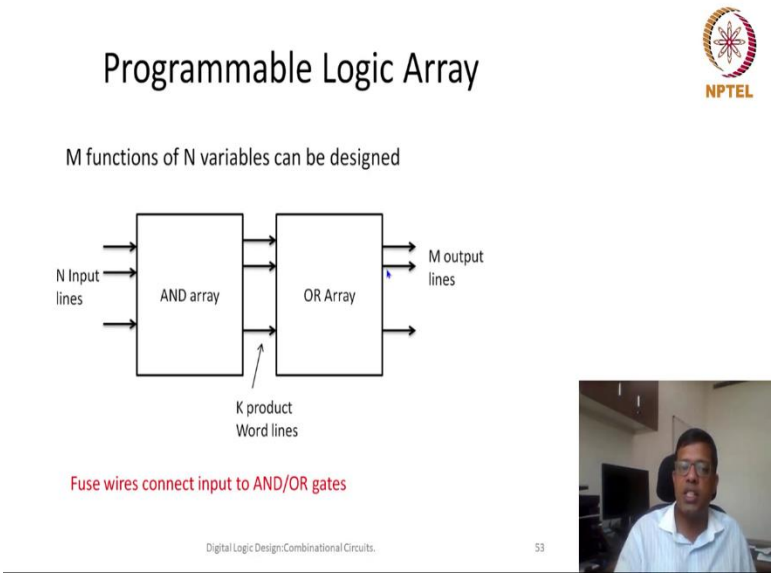


Now, all these programmable hardware devices or logic devices they are typically called PLD programmable logic devices. They are various in nature different type of programmable devices

are there. They are all of them are programmable but by different means there are various examples. So we will see couple of more here other than memories, so other than ROM's because ROM would have certain challenges.

We can have PLA's programmable logic array, it could be CPLD complex programmable logic devices or FPGA field programmable gate arrays. So we will see all of these one by one.

(Refer Slide Time: 25:36)



So, in PLA what would happen PLA's is a standard AND OR combination, so we have seen that any N input logic can be designed using 2 level logic sum of mean terms or we can say sum of products. Now, what is this product? Product means it is a AND gate and sum means it is a OR gate. So, here in this programmable logic array there are k AND gates. And all of these AND gates are connected to these N input wires at actually 2 N, so because each input would be available in original as well as complemented form.

So, such K AND gates would be there and all these K AND gates would be connected to my M output line so basically M OR gates would be there. So, each OR gates can be connected to any of these K products depending on which all product it is connected to it will create different function. So effectively we would be able to create M different functions assuming all of them are using same N variables.

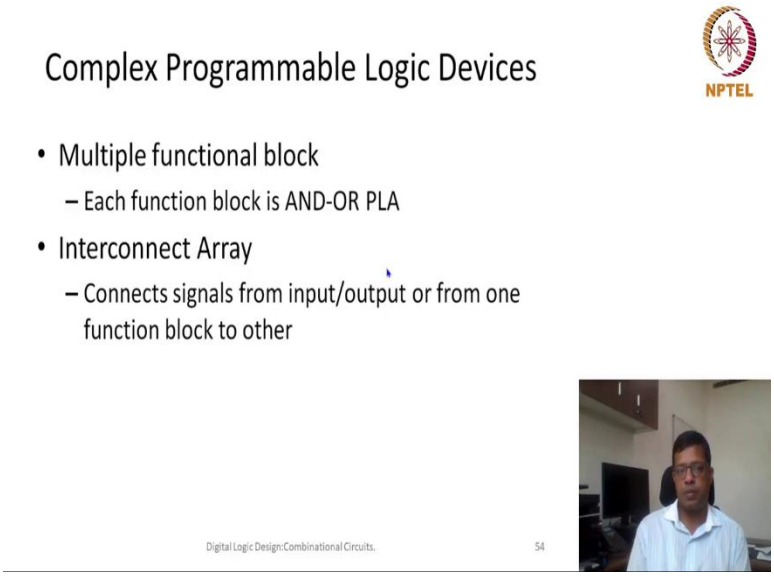
So some of you were asking and thinking that why we are taking inverted as well as original input as available. So it is primarily because of this programmable logic array. So in early 80's

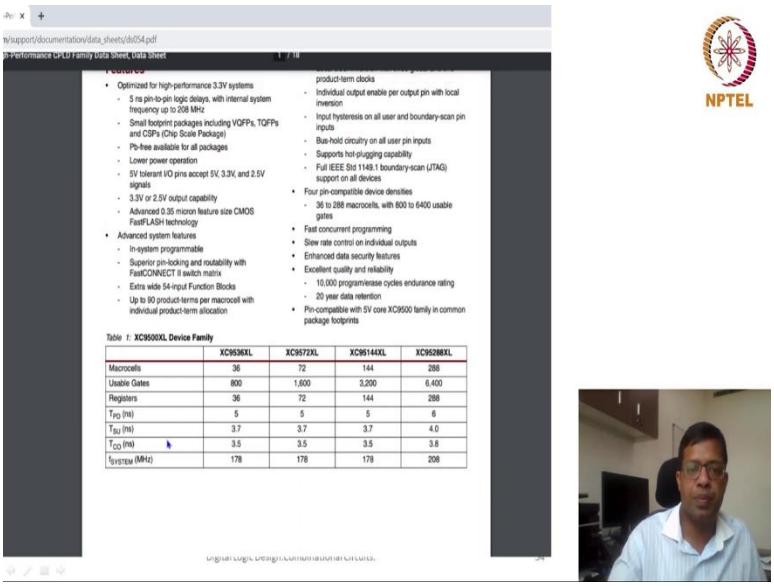
when people have started thinking about programmable logic then they come up with this idea and in this idea so because the cost of inverter was used once. So what they state that, whatever N input lines were there they created the inverted of these input lines also and assume that in the AND array all of them are available as input.

Now, each of this AND gate can be connected to either the original input or the inverted input. Now, which would be the actual lines which would get connected to it, so there are some fuse wires here also and using these fuse wires the connections whatever we would like to connect, so let us say out of these there are 3 inputs A B C. So out of A B and C we would like to connect one particular AND gate with A and B bar,0 so although there are 6 inputs which could be connected to and this 1 particular AND but only those two would be fused otherwise rest would be open circuited they will not be connected.

So, similarly there would be K such AND gates or K such product terms and using those K product terms again there would be fuse wires here it would choose that which particular value should be connected to each OR gate. Now, what should be the value of K? Ideally it should be equal to 2 is to power if there is N inputs there should be 2 is to power N product terms. But because of optimality it is usually kept below 2 is to power N so that because not all the product terms would be required in all the functions. So, this is how programmable logic array works so it is generic and AND OR array gates and the fusing is or basically connections is done using fuse wires.

(Refer Slide Time: 29:33)



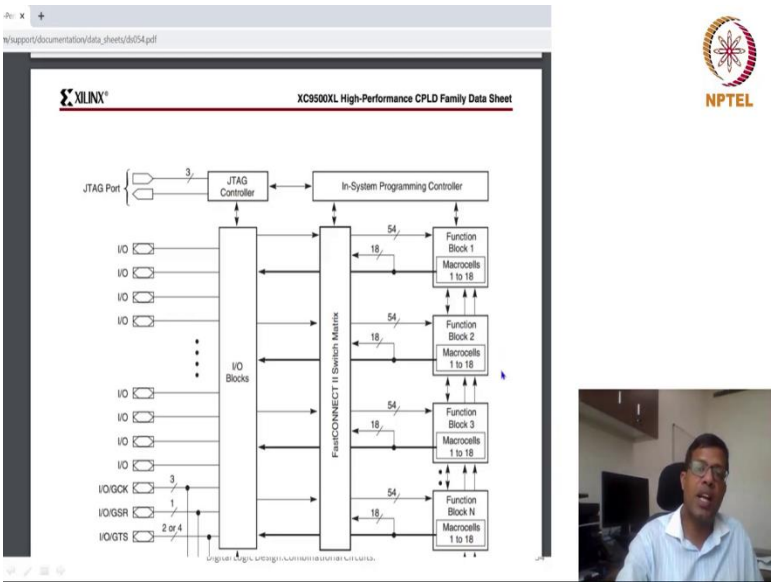


Now, the next one CPLD complex programmable logic array. So, after people have designed PLD's, PLA's so they wanted to make something more complicated. In PLA you can have a input and 1 function, so the output of this function cannot go to some other as a input to some other function. So, basically this is the intuitive idea behind a CPLD's, so in CPLD'S they are multiple function blocks each of this function block is actually a PLA and there is a interconnection array, so in this because of this interconnection array the output which came because of 1 function block can also be given as a input to other function block.

So using this what you are effectively able to do, you are effectively able to create a multiple level logic or a logic where output depends on not only the inputs but the process inputs.

This CPLD is coming from a company called Xilinx its name is XE9500XL high performance CPLD so this is a family which mean that there could be multiple specifications, number of macro cells could be different, number of registers, number of input etcetera could be different.

(Refer Slide Time: 31:01)

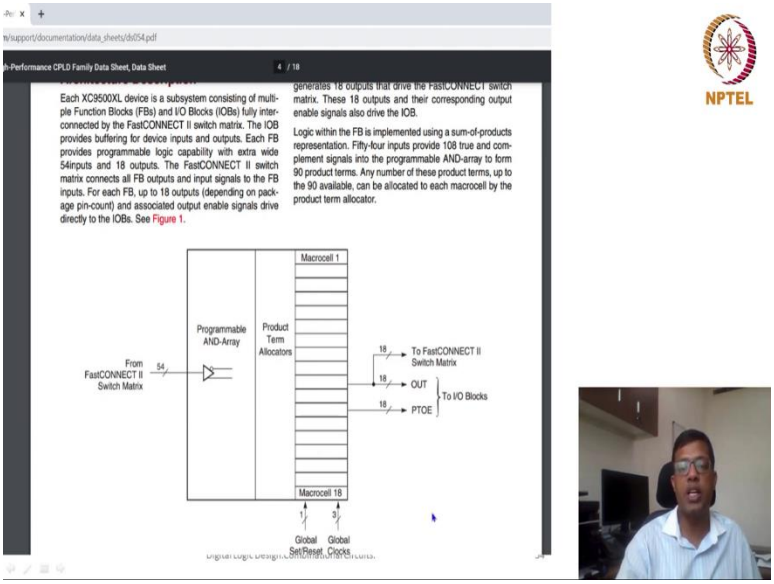


So the overall architecture, overall design of this CPLD is therefore function blocks and in each function blocks there is a possibility of 18 micro cells, each micro cell means 1 PLA. So, and there could be 18 outputs of that PLA, so which essentially means that you have 18 outputs and number of inputs are also 18. Now, it is written 54 then why I am saying 18, because it is 18 plus 18, input is available in original as well as complimented form. Inputs are available from here, so from here this interconnection would be able to give the input to all these function blocks and there it can again generate 18 outputs.

So, these 18 outputs are again given back to this interconnection block so that this 18 one of some of them can also be given as a input to other function blocks. So, similarly all these 4 function blocks are connected now I can create more complex logic which can take all these different 18 inputs and it can also do some processing in between from it can pass the output from 1 function block and it can give it to the other function block.

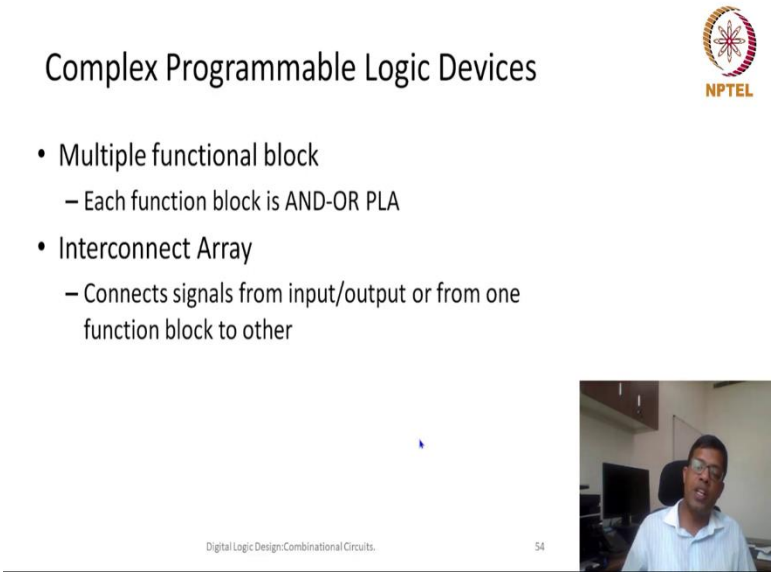
So, this way because it was complexed at that time so that is why it is called complex CPLD or complex programmable logic device.

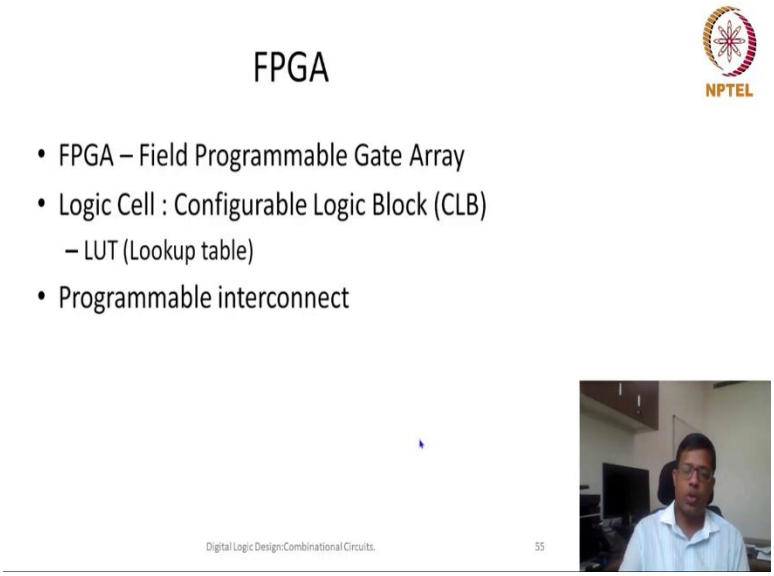
(Refer Slide Time: 32:45)



Let us quickly look at this 18 micro cells also, so you have 54 inputs and each of this product of terms are there so you can have this hand AND OR cells are there. So, these are the product terms and after that you have all OR cells which will generate different outputs so there 18 outputs which are possible because of this.

(Refer Slide Time: 33:07)





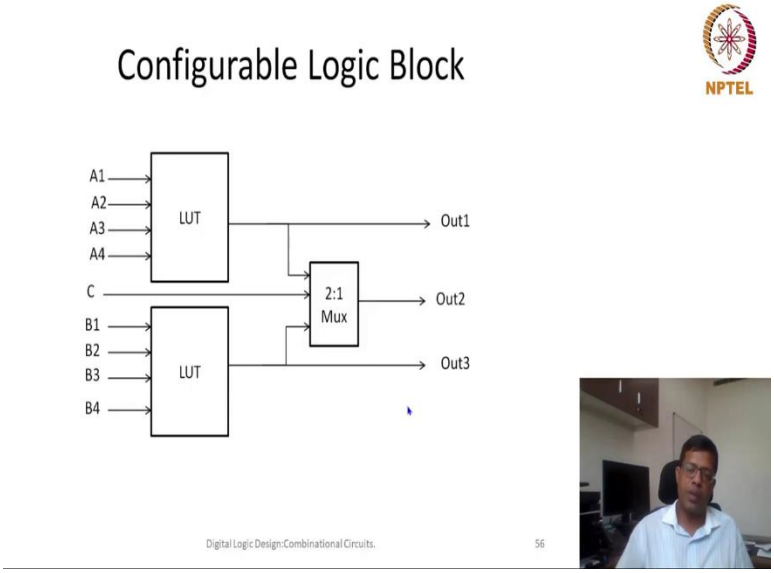
This was not even sufficient that this CPLD were able to make something more programmable but even more programmability was required so then we restored to something more like which is field programmable so that we can program it again and again field is usually electric field here. So, basically this is programmable using electric field or voltage or current.

Now, here you have you call it gate array because there is some basic logic which has been repeated again and again. So, here the basic unit is configurable logic block people call it with

different names at different proprietary hardware. So but the idea is that some particular cell is created which has been repeated again and again which has been used again and again.

So this logic cell also called configurable logic block will usually have some memory look up table inside it and there will also be a programmable interconnect so any of the CLB can be connected to the other CLB. So this is another way of programming a logic or a logic which can program using FPG also.

(Refer Slide Time: 34:31)



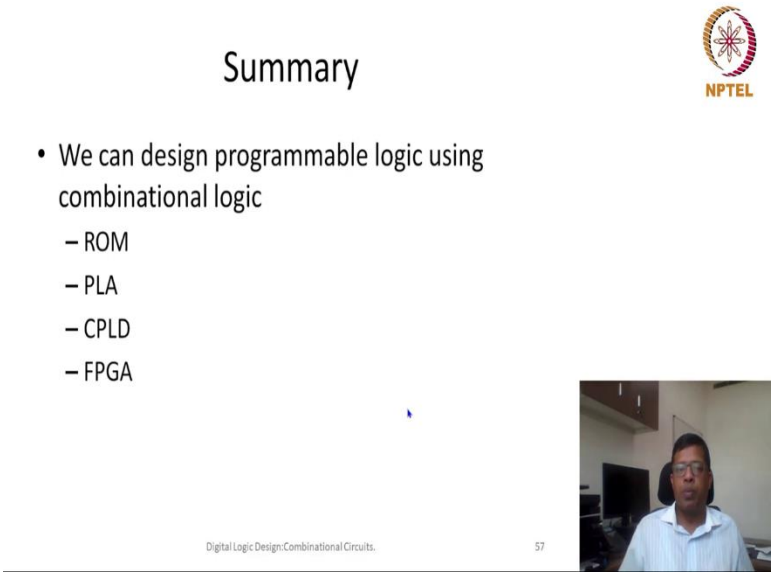
A typical CLB looks something like this that let us say there are 2 look up tables, so look up table is essentially combination of your memory as well as your multiplexer so that it takes 4 input but it stores the output. So whatever is the content of LUT so let us say if it is 4 input and LUT then there would be it would internally have 16 cross 1 memory. So that 16 cross 1 memory would store the truth table content of a function which we would like to execute plus there would be a mux which would give output as on them as a output based on whatever is the input.

So this is 1 CLB block which has 2 LUT's and this 2 LUT's has both of them has 4 inputs and there is mux here. So it can have like 2 or 3 different outputs, so this CLB can be used to implement 2 functions of 4 input each can also be used to create 5 input functions so 2 input from here and 1 input from here. So 5 input functions can be created using this CLB and some selected 9 input function are also possible to be created from this CLB.

So the idea is that you will feed the content of your truth table in your look up table in the memory and this lookup table will give you the output corresponding to 1 function whatever we would like to design or program.

So this way this FPGS can also be used as a programmable logic. Now, with this we will close this lecture and we will learn about this configurable logic block once we had, we will also see how the sequential logic will work and then we will have one more lecture explaining how this programmable can be created.

(Refer Slide Time: 36:37)



So in summary what we have done in today's lecture, we have seen how a programmable logic can be created using generic combinational circuits, so we have seen 4 methods read only memory, PLA's programmable logic arrays, CPLD's as well as FPGA. So any of these methods can be used but as we are going more and more complex, so these days mostly FPGA is used as a programmable hardware solution. Thank you very much.