Digital Circuits and Systems
Prof: S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture - 31
Programmable Logic Devices

(Refer Slide Time: 1:55)

In the last few lectures we have been seeing about the design of the combination of logic using MSI Medium Scale Integrating circuits. We used examples of multiplexers, decoders and saw how to map a given specification given requirement into hardware using these components. We said that in terms of the count of the number of ICs these are efficient because of the small number consequently leading to smaller size and lower cost, higher performance and lower power deification. There is a limitation on these ICs in the sense of the number of gate functions these circuits can give because we just roughly defined Medium Scale Integrated circuit the circuit with about ten to hundred gate functions so the circuit becomes larger in that. the definition is all right they are called large Scale Integrated circuits but leaving the definition apart the circuit becomes large it's not easy to map that during that during the heuristic mapping intuitive mapping that we have been doing for the multiplexers and decoders because there is no systematic procedure conceptual procedure so the usage becomes inefficient. Mapping more and more hardware into a given requirement becomes more and more difficult in a best possible way that is one thing. second thing is when you have circuit of that scale large circuit with large number of gate functions it will be nice to be have several uses for that, I would like to use it for different applications.

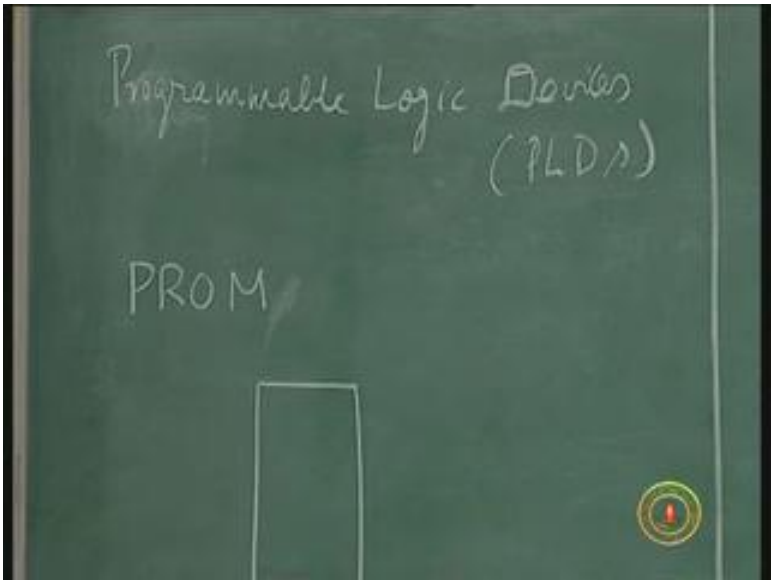
I may buy it and use it for one application and later on I may change the application, I may not drastically change the application but I may change some of the specifications

that you get, I may change some requirement, some of the specifications may change in which case I do not have to read do it I may go about and get another IC and then do different designs. If I can make some changes in the design and use the same hardware this concept is called programmability. Programmability is a very important concept in digital design. That is why you get microprocessors in computers. Today we will say Programmable Logic Devices PLDs is what I am going to talk about. Programmability means when a given hardware is used for different applications by changing the design in computer. You know that the same computer can be used for different applications by changing the program, you are re-programming it. You are not familiar with microprocessors but microprocessors are nothing but computers again. These are also processing units just as you have computers. in face a computer has a microprocessor, the processing unit of a computer is a microprocessor.

The term micro came because it is a small circuit because it is one single chip processing unit, that's why it is called a microprocessor. So if I have a microprocessor with an expensive piece of hardware I would like to use it for different applications by re-programming it, programming for different applications. These are some of the things I would like to have so in the same way I would extend this concept of programmability to even less extensive hardware not necessarily to that of computers or microprocessors but I may have a piece of hardware and I want to configure it for an application. I may want to change it for different applications. Or, even in the same application if I want to reconfigure it for different specifications then I should be able to do it. This programmability helps in using the hardware repeatedly, the reuse of hardware, saving cost and designing an efficient system. We cannot always get the best possible design in the first iteration, you may have to iterate it several times before we can get it perfect in design.

So, concept of programmability becomes important and added to fact that when a large system is there like a Large Scale Integrated circuit mapping is not an easy task it is not a trivial task so there again we have the programmability because I may want to map it and then want to change it and different configurations remain. We want to try different possibilities and finally fix the configuration. This software is something which all of you are familiar with. That is, when you write a program for a computer then you call it software. Actually the word software is used by so many people in so many different senses and in so many different ways and they really do not know the difference between them the definition of software and hardware etc.

(Refer Slide Time: 12:45)



So, in between there is something called Firmware, have you heard the term Firmware? When you take a computer a computer is a hardware you have application programs running in it, your word processing, even the operating systems, windows 2000 these are all software but then some of these subtle changes I make, specification changes, some of these things are not changed very frequently, I change it I put it there and then if I have to change it I may have to redo it, this is not like software where for the same computer for the same day I can use different software for different times. In today's computers I can run several software in parallel, multi-tasking they call it, multiple processing, processing multiple programs. Whereas in Firmware configuring the computer for a particular use requires certain programming and the programmability that I am talking about comes under that category, that's why I am introducing the word Firmware because this is not the program that we are familiar with in the software sense.

Programming is not programming in the software sense that we are familiar with in day to day point of view. When somebody says software you understand something, you understand a word processor or a database application. I do not mean programmability in that sense here. the programmability required for reconfiguring it for example a computer will be reconfigured for certain applications, certain class of applications, this comes with the system program but I may change it, if I change the firmware.

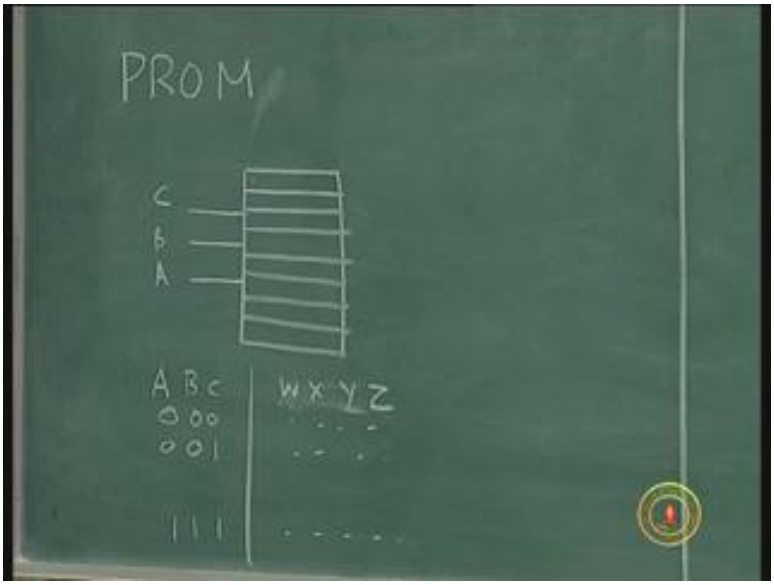
Firmware defines the usage the available resources of the computer on which you write the software to run for different applications. So when I say programmability I mean programmability in that sense of firmware here. So I may have a piece of hardware in which I can change a part of this either replaced by another hardware a hardware which has a program I throw it out and put another piece of hardware which has another program or use the same hardware in which I will erase the program and then put another program, so this level of programmability is the Firmware. This is a subtle point.

Now as I said we already made a case for the reason for programmability and programmability in a micro sense here not in an application sense or a macro sense. They were originally devices known as PROMs. There are many devices which are programmable logic devices. First one is a PROM, PROM stands for Programmable Read Only Memory and ROM stands for Read Only Memory so Programmable Read Only Memory is PROM. As I said we are interested in combinational logic and sequential logic in this course, it is a basic digital design course first level course in digital design. I am not interested in the computer or microprocessor programming languages and programming skills and all that.

So to that extent how is this ROM going to be helpful in programming in Firmware. Now, basically all these designs either it is combinational design or a sequential design where a combinational logic is used for steering we have a truth table which has to been implemented, we call it transition table we call it state table or whatever it is, that is the truth table which has to be implemented with input and output relationship and we simplify using Karnaugh Maps and then get the minimum possible hardware. Instead I want to implement the truth table directly in hardware that is a Read Only Memory.

A Read Only Memory is nothing but a hardware truth table. If I can etch the truth table in hardware in silicon then it becomes a ROM because each of the possible combination of inputs should produce the corresponding output and that is the truth table. So the ROM will have the three inputs ABC and each of these inputs they may have certain value stored in it for that particular combination of inputs and these are the outputs, there are eight combinations 1 2 3 4 5 6 7 8. So if I want to have a truth table with three value ABC and three value output XYZ or four values of output W X Y Z for ABC input is 0 0 0 let us say this is something, 0 0 one that is another combination and 1 1 1 that is another combination, I can etch this in this silicon, this is a truth table I can implement it using reducing the Karnaugh Map and then using gates.

(Refer Slide Time: 14:00)



Now instead of doing that what I am going to do is take of piece of hardware called the ROM Read Only Memory to which I will give the input as the three variables the current value of the three variables each of them having 0 or 1. That means I am having eight possible input combinations and corresponding to that combination the required output should come out of this. Others supposing the input combination is one 0 0 the output should become 1 1 0 0 so as soon as they make one 0 0, let us say this is that is 0 1 2 3 4 value 1 1 0 0 would be written here which I will be able to read off. This is the Read Only Memory and there is no program here.

I just give the value I get the truth table I etch it in hardware as you will. I am just exaggerating that, you don't take a chisel, silicon, rock and do that, imagine it to be a hardware, hardware truth table I am writing in paper I give you a chisel and a piece of silicon you etch it so that is a ROM Read Only Memory. Now, how does it help me to design a combinational logic? This is the combinational logic A B C (Refer Slide Time: 15:45) put W X Y Z output that is what I want to implement it could be a combinational logic function or a steering logic or a sequential logic function, I use this ROM feed A B C as inputs and the corresponding values are etched in the outputs but then it is fixed, I can't change it.

The programmability concept does not come in there, it is the same as the original description now. So for each truth table I need to put a new etch I need to go and get a new slab of silicon. I want to use the same hardware and use it for different applications that means there must be some different programmability. So if there are three inputs in the function all the eight combinations of A B C can occur there is no other possibility so inputs are common. Whatever is the application whatever is the variation in the outputs there are only eight combinations of input possible 0 0 0 to 1 1 1 that part is common. what is different is for each combination of the input the corresponding outputs are different for different systems, different truth table different steering tables, different combinational logic functions will be implemented with different outputs for the same set of input combinations.

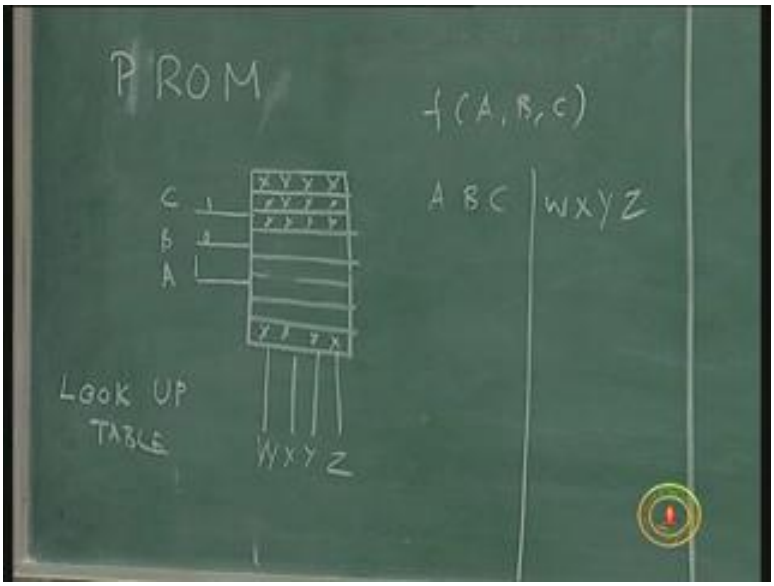
So instead of making it fixed etched hardwired truth table I will add P to it so introduce this programmability here I will write as required in my truth table every time the new value that is the programming aspect of it. So instead of fixing 1 1 0 0 0 I will have these four xs which will be changed to my requirement. So I will now get a new truth table for the given application in that truth table for each of the combinations what is the value of W X Y Z and those valuable for W X Y Z I should put in that position for the 0 0 input. That procedure of putting the different values of w x y z for different combinations is called programming. This is programming in a sort of a crude sense.

All of you are familiar with C language now being taught as interaction to computing. So we do not do that type of programming here. I only change the programmability in the sense I can select rather it is a selectable design. Programming is a word, programmability is the change and because it is changed it is called programmable, it can be changed.

So what is new value of W X Y Z? For each of these combinations of A B C it will be decided on that particular truth table that you are going to implement which will depend on the input output relation that you want to implement for that particular system. So how are you going to make it possible? How are you going to make this programming possible? So now it's a two step process. I used a single piece of hardware where I have three inputs and four outputs with the fixed values ROM, ROM is a hardware truth table but it is a fixed truth table.

A fixed truth table has another name called look up table. That means like a multiplication table because you can only look at the value you can't change the value twelve times twelve is 144 can you change it? No, but you can look up if you do not know, of course you don't know so you look up all the time.

(Refer Slide Time: 19:55)

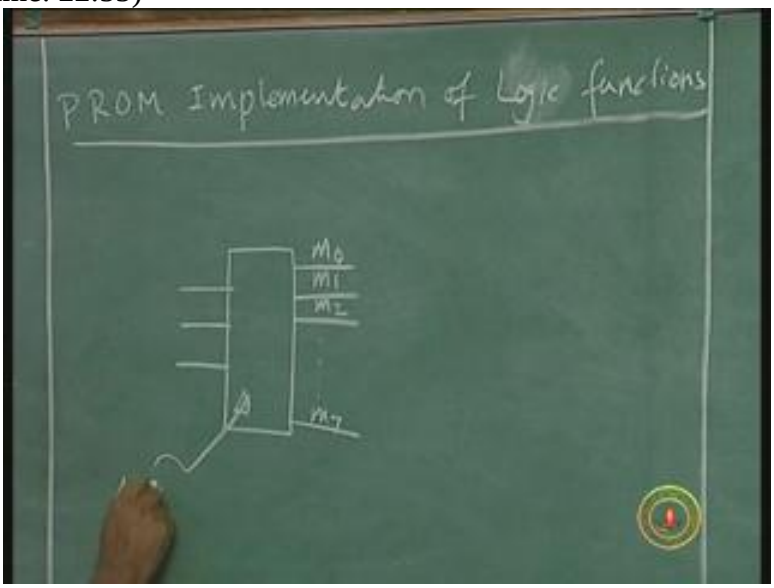


Now-a-days you use computers for everything. That is finding a student using some addition with three binary numbers wasting 15 minutes on that programming it to get the values because you are used so much to computing. so fixed truth table is a look up table where there is no programmability but programmability has to be there for me to implement different functions using the same piece of ROM. Such a ROM whose content can be changed for different applications is called a PROM, PROM stands for Programmable Read Only Memory that is the first programmable logic device we are using. That is why I said PROM programmable logic device many programmable logic devices we are going to discuss, and the first one of them is the Programmable Read Only Memory. Now I have to tell you how I use a programmable read only memory to implement different logic functions. That means my ROM is going to consist of two input values.

PROM implementation of logic functions:

now in the first part of the ROM I need to get all the min terms because in the truth table the left hand side is common whatever is the output whatever is the system the left hand side of the truth table is common for the three inputs and it has only eight combinations 0 0 0 to 1 1 1, four inputs sixteen combinations, five inputs thirty two combinations. So that is the fixed part. How do you get it? First part is a bar B bar c bar, second is a bar b bar c then A bar B C bar etc and finally it will be a b c so that I should get here and a bar b bar c bar is also called m0 min term 0 and what should be inside this? There should be AND gates inside this, that's all. I want a bar B bar and c bar I want three inverters and one AND gate with three inputs. If you want a b c I need three input AND gate A B C.

(Refer Slide Time: 22:55)



So this is AND they call it AND array because there is a large number of AND gates to provide all these mid terms, these are the inputs. Now my programmability comes only in the second half. How do I combine my min terms for different applications? For different applications there are different truth tables, different transition tables, different steering logic tables and the outputs will be combined differently.

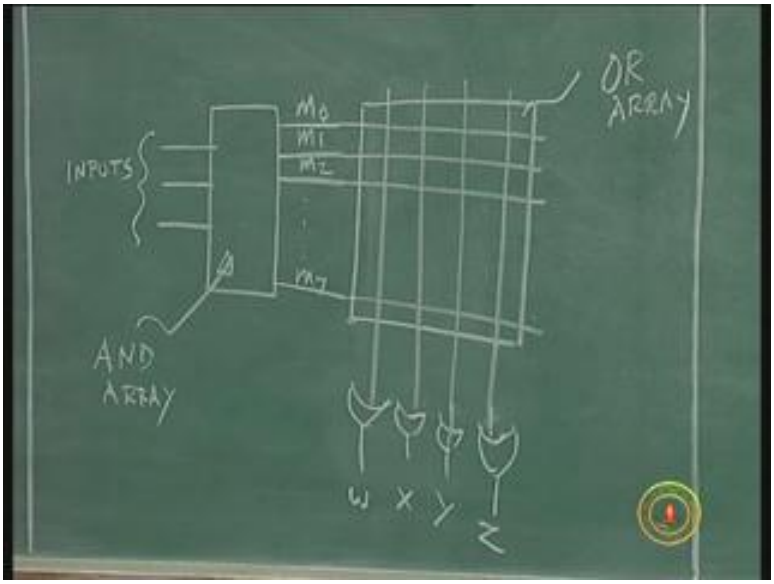
For example, I may need m_0, m_1, m_2 in one application and in another application I need m_0 m_1 and m_7 so I should be able to select some of those min terms and combine them in OR gates. The second part is all the eight min terms are available out of which you have to choose which of the min terms are required to implement a particular function W or X or Y or Z so that will consist of OR gates. So how many OR gates must be there depending on the number of outputs required. There are four outputs required W X Y Z four OR gates in which all the eight inputs will be fed so I should be able to retain some of them and disconnect some of them.

So I will draw this symbolically these four OR gates with eight inputs each. Symbolically I will say that there are four outputs W X Y Z which is nothing but a combination of this, this OR gate is only one input symbolically and each of these OR gates have eight inputs

m_0 m_1 m_2 m_3 m_4 m_5 m_6 and m_7 out of which I will retain those which are required for my particular output based on the truth table and remove those which are not required in the output. So this is W X Y Z, this an OR array but this is programmable in the sense I can select which are those m0 m7 which will go into input OR gate and which of them will not go into the OR gate. Those which will go into the OR gates will depend on the function. After having looked at the truth table you decide may be m1 has to go, m_7 has to go, m_3 has to go, m_2 has to go and rest of them will have to go so only retain those connections.

First I have the provision to connect anything and how many of them will be used depends on my requirement so this is an OR array (Refer Slide Time: 26:16). This is fixed for all designs so I will call it a fixed AND array. This is programmable, each for different applications I can have it so I will say programmable OR array. for example, W is m_0 plus m_1, m_3 plus m_7 as an example then I will have to retain in this connection so whenever I find a dot connection is retained and whenever I don't put a dot you assume that the connection is not put.

(Refer Slide Time: 27:00)



There are eight inputs waiting to go into this OR gate, I retain some of them and terminate some of them. For whichever I put a dot there a thick dot assume it is connected and for whichever I do not put a thick dot assume it is not connected. we have to have some understanding, see it is only some hardware finally and not the board so whatever you write on the chalk board it is not going to work we have to make a chip, we have to make an interconnection so m_0 m_1 m_3 and m_2 m_4 m_6 has not been connected.

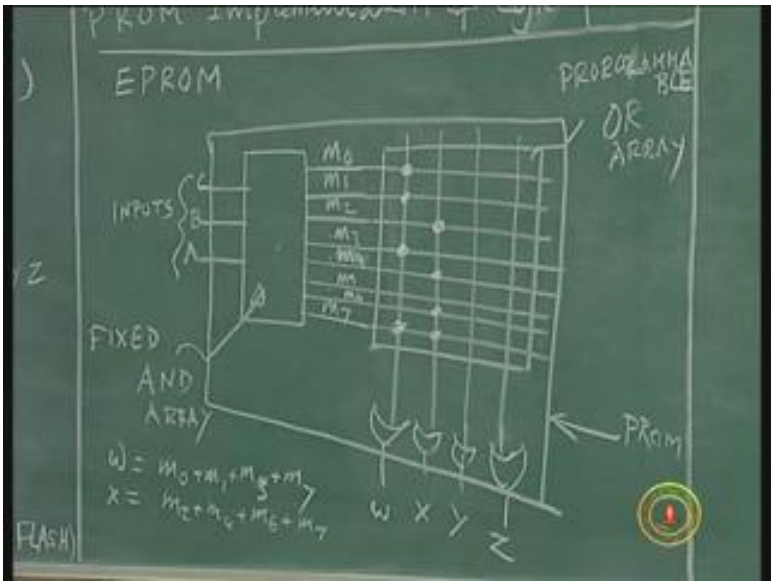
Similarly, for X I have different combinations X may be m_2 plus m_4 plus m_6 plus m_7 so it may be m_2 m_4 m_6 so it is only a symbolic presentation. Now the programmable device comes into picture. This is a programmable device, these are my inputs that are available and I don't have any control over what is inside this. And up to this it is a PROM (Refer

Slide Time: 29:01) in which I have fixed AND gates and produced all the min terms required and the programmable OR array in which I can retain such of those min terms for each of these outputs and not connect such of those inputs which are not required for that particular output.

How would you do it physically? There are different technologies. I am not going to bring in technology here we will learn this in some other course. Suppose there is a fuse like a fuse wire at home, have you ever seen a fuse wire? So you can have a fuse wire which can be snapped if you want to. Those who want to retain those connections you leave these fuse wire intact and if you don't want a connection you snap it that means there is no connection that means it is called fuse technology anti-fuse technology, so it is fusing or anti-fusing. But then once you snap it you can't get it back.

For another application I want to change it I have already snapped it in my previous application so that is not a good technology. So there is a charging and discharging. We can have a charged packet keep it when you want to the connection and discharge it when you do not want the connection. Again we can charge it. So this are all different technologies called programmable devices which you will probably learn in your semiconductor devices and technology courses.

(Refer Slide Time: 36:30)



Those who are doing VLSI Technology will do this in more detail and everybody will be doing a semiconductor devices course the Solid State Devices in which you will learn some of these things. So let me not get into that domain here but suffice it to say that there is a provision by which I can retain some of these connections and remove some of those connections and re-program it for certain other combinations also. Because once I have retained and then remove it does not mean that it may be useless. Suppose I want to change my configuration I want to make an error correction suppose if I find this error this is small one or two mistakes in the connection I can always redo it so I should be

able to erase it to reprogram it, it is called a Erasable Programmable memory EPROM, ROM is the truth table etched in silicon, hardware truth table, PROM is a truth table whose output can be changed as required and Erasable Programmable Read Only Memory is a Programmable Read Only Memory whose programming I can erase and reprogram it for different applications or for changing a configuration or making a correction. How do you erase it, there are technologies. You can erase it by shining Ultra Violet light called UV erasable Ultra Violet light erasable technology or Electrically Erasable EE, so I can have ROM Read Only Memory, PROM is Programmed Read Only Memory and EPROM is a Erasable Programmable Read Only Memory and UVEPROM is same Programmable Read Only Memory and if you use the erasable technology using Ultra Violet rays then it is called UVEPROM or you can do Electrically Erasable EEPROM Electrically Erasable Program and there is one other technology called Electrically Alterable technology EAPROM programmable memory there is a difference between electrically erasable and electrically alterable I can go and change something only, erasing means you erase the whole thing, I have a slate in which I erase it and start all over again.

Ultra Violet rays will erase the whole thing electrically erasable I can erase the whole thing whereas electrically alterable I can selectively go and alter, if I want to reprogram it or especially when I want to correct a bug I will go and alter it like surgery nowadays, earlier they used to cut open and today they can point a laser beam and make a correction so it becomes electrically alterable. Electrically alterable is also known as flash memory. Have you heard of this term flash memory people use? It comes in popular magazines, science sections of newspapers and so on.

Leaving all this apart I am having a programmable feature in my ROM by which I can connect those inputs which are required into OR gate and those inputs which are not required in OR gate I will not connect them, I can change it by erasing it and alter so all that is possible. So this is how I have a programmable solution for my logic function. Now I have a three input four output truth table to be implemented, I take a ROM with seven min terms connect my inputs A B C here and get seven outputs so these connections are made by programming so that w will represent this and X Y Z will represent this.

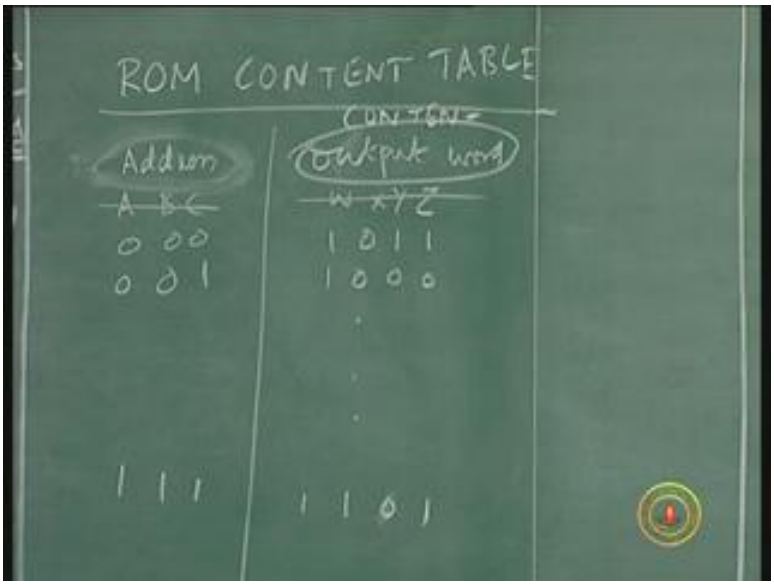
I am not going to tell you how am I going to go into it and do it as is said the technology of programming and erasing I am not going tell you because it is not a part of this course it is too much of technology. Similarly how do you do it programming in a lab you do not open it and then look for the OR connection and put it, you don't do that so by externally injecting signals currents we have to program it and erase it so there is a device called PROM programmer.

You insert a device in this PROM programmer and type in the word, for each word each address this you call addresses (Refer Slide Time: 35:46) so these three inputs will generate eight addresses starting from 0 to 7 and for each address what should be the value of W X Y Z will be given so I go and type the first address and write what word should be there. What word should be there depends on whether W is 0 or 1, X is 0 or 1,

Y is 0 or 1 and Z is 0 or 1 then I go and change my address to 0 0 1 there again I will say what is W, what is X, what is Y and what is Z this is called programming so I have a little equipment called the programmer PROM programmer in which I insert this little device PROM and then first type the address and the corresponding data in it. So these are called addresses, the ROM content. You make a ROM content table so address is nothing but the concatenation of the A B C inputs and output word. This are not any different this is only a jargon here so here I have a b c and here I have w x y z.

So 0 0 0 address will have whatever 1 0 1 1 as I say and then I will say 0 0 1 and something like this (Refer Slide Time: 37:20) so the art of putting this is programming, the art of putting this ROM content table into the ROM physically electrically rather is called the programming aspect that can be done by a simple device called programmer wherein you insert your ROM and type in the address and corresponding data so this called the address and this is called the content. So A B C has no meaning here it is only for your mapping. Address has A B C and output has W X Y Z.

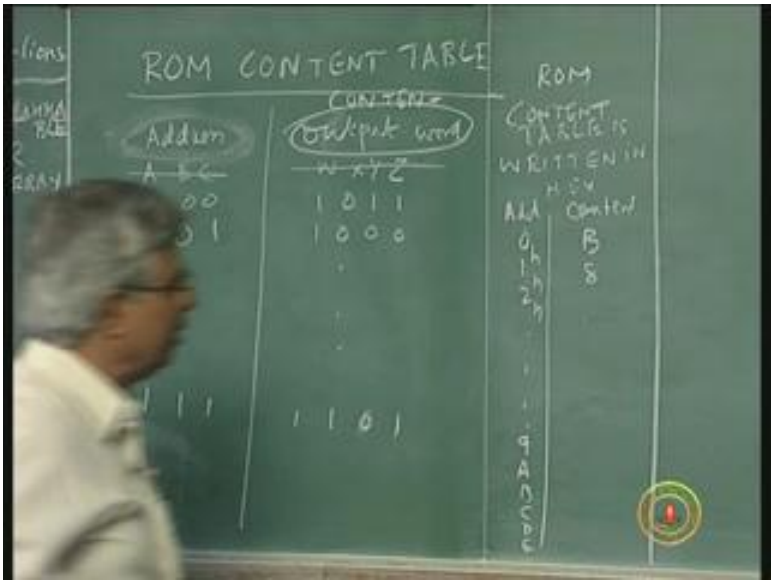
(Refer Slide Time: 38:24)



You do not write in binary but you write in hexadecimal usually. Usually ROM content table is written in hexadecimal form because remember these are all very tiny examples that I am doing in this class and I don't think anybody will be doing a seven word ROM, ROM with only eight words and four outputs. Usually in the big systems you may want to have several inputs and several outputs so if I start writing 0s and 1s all over it gets too confusing to handle so usually write in hexadecimal.

So this will be in hex so this will be 0 1 2 ROM content table in hex which is 0 1 2 address and you have the content, so corresponding to that address what is the content, and all these are hexadecimal 7 8 9 10 and after 9 it is A B C D E and for example this is called 1 0 1 1 so how do you refer 1 0 1 1 in hexadecimal, so after 9 instead of 10 you get A, and 11 is B, 12 is C, 13 is D, 14 id E and 15 is F so this is B, second is 8 and so forth.

(Refer Slide Time: 40:50)



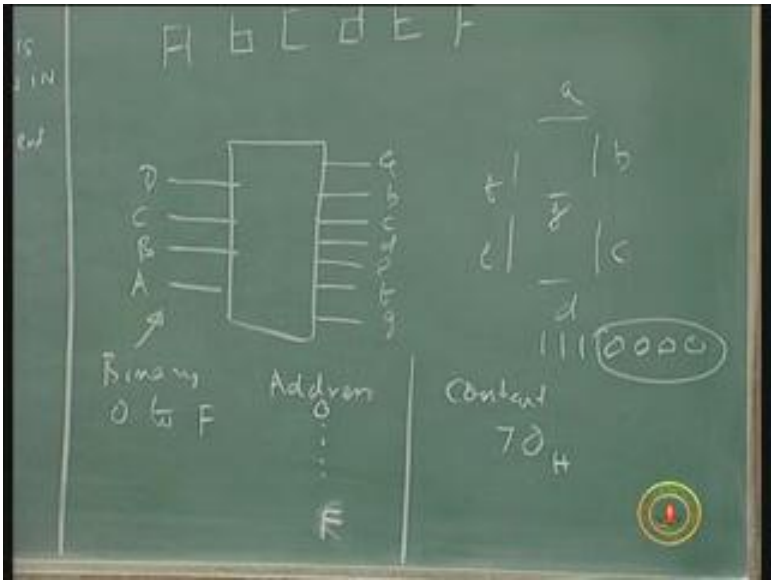
So I will give you an exercise now. Do you all remember that 4-bit binary to 7 segment converter we did? BCD to 7 segment converter decoder, we see the 7 segment decoder we did earlier, take it and put in a ROM, you get a ROM content table for that exercise, don't make it BCD but make it binary.

The problem is, I have a 4-bit input in binary that goes from 0 to F values and the output will be seven segments so I will call this a b c d e f g, we draw a table and 0 1 2 3 up to F it should show so 0 1 2 3 4 5 6 7 8 9 so it is A b C d E F for this you have to do a ROM design. One single ROM can do this job, you need four inputs A B C D and seven outputs A B C D E F G and then for each of this address 0 0 0 0 you have to find whether A should be the segment, you should know the segment allocation.

So for each of these characters you want to know whether A is on, B is on, C is on, D is on, E is on, F is on, G is on. So you have to know which is on, on is 1 and off is 0 write that table and then that will be the ROM table so first write it in binary and then convert it into hexadecimal.

In hexadecimal 0 to F is the address, so I want you to give me the final answer like this 0 up to F address and content would be seven bits. How do you write a 7-bit hexadecimal? It is the first four bits LSB one hexadecimal digit, 7-bit table number. For example if you have numbers 1 1 1 0 0 0 0 this is 0 (Refer Slide Time: 43:43) and this is 7 then this is called hexadecimal 7 0. Like that for each of these things I want you to get the correct hexadecimal.

(Refer Slide Time: 43:55)

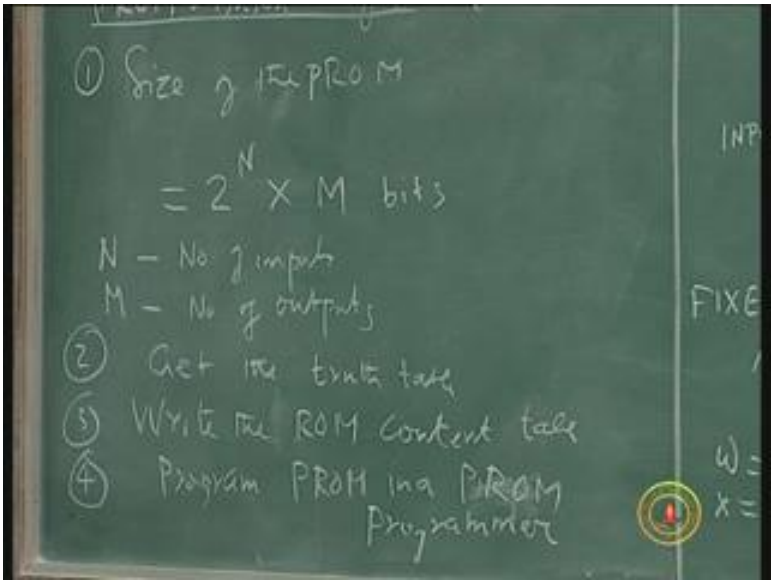


So, ROM design is one of the simplest designs possible. The only thing you need to know about the PROM design is once the problem is given you should ask for the correct size of the ROM i.e. how many inputs and how many outputs. For this problem I needed three inputs four outputs, for this problem (Refer Slide Time: 44:22) I need four inputs and seven outputs. So the first thing in a ROM design is you ask for the correct size.

The size of the ROM is given as 2 power N times M where N is the number of inputs and M is the number of outputs. The three inputs will have eight rows and each row will have four outputs W X Y Z so 2 power N times M, 2 power 3 times 4. What are the units of size? Every size should have a unit right? They are bits. Byte is nothing but 8 bits, so here it is only bits. It is 2 power N into M bits. Therefore once you have the size of the ROM get the truth table ROM design it is the PROM based design, so wherever I write ROM you rewrite it as PROM because ROM is once only like your multiplication table which is fixed and ROM comes from factory.

Supposing you have an equipment in which everything is tuned and they don't want to meddle with that, they will put a ROM so you cannot erase it and you cannot change it, factory programmed erase call. A ROM when it is programmed in the factory you will not be allowed to alter it. supposing I have some program which comes in the system, the system program, as soon as you switch on the power of a computer so many things happen before you can really start typing your keys. For all those things I do not want individuals to do it because then everybody will be doing it in a different way, it will work and it will not work and you will blame the company not designing a proper computer.

(Refer Slide Time: 49:13)



So they put that in the Boot strap ROM the boot ROM, ROM is in which they put all those initial program and for some of those companies that is the proprietary item because you can always put whatever you want a junk and then you can claim that your computer is junk so the patent comes from the original ROM in which they are able to do everything at the system level.

Similarly in a music synthesis as soon as you switch on of course you may want to do your own music but then there will be a standard music "Yankee Doodle" or something like this it starts when you switch on that is again programmed in the ROM and they wouldn't change it then. "Yankee Doodle" are all these American Rhymes. In a calculator the sin table and cos table is all programmed. If I ask you to put your own sin table and cos table then it will be a catastrophe. When you say sin and 37 degrees it promptly becomes a number because it goes and looks up the value of 37 in the ROM table look up table it is and it is not a programmable table and the value of sign is coming out and you use it on your computers.

So PROM you get the size and then get the truth table and write the ROM content. You don't call it PROM content but you call it ROM content even though it is PROM, the next step is you store get out of a programmer program PROM in a PROM programmer. The industry people who are doing this for a long period will say burn the ROM. Have you burnt it? These are all the pet jargon they use. Burning a ROM is putting the program in it, actually they don't burn anything. We are assuming that it is a fuse which is flowing and then it is going up in flames so this is all imaginary. putting a 0 or 1 there is called programming called burning in colloquial terminology for those designers who do it often it is burning, they ask, have you burnt the ROM.

If your ROM is wrong then you will have to burn your exam papers. Make sure that it does not happen to you. So this is a two-fold treatment today. I wanted to introduce the

concept on large scale integrated circuits and reasons of programmability and also the ROM as a technology ROM, PROM, erasable ROM, erasable programmable ROM and all that. There are other programming devices also. We did a simple example and I asked you to do one more example. So the next programmable device is called Programmable Logic Array PLA. We will talk about it in the next lecture. Programmable Logic Array in which a similar programming feature is built in but how is it different between ROM, PROM and how is it more efficient than ROM we will see in the next lecture.