Digital Circuits and Systems

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

Lecture –4

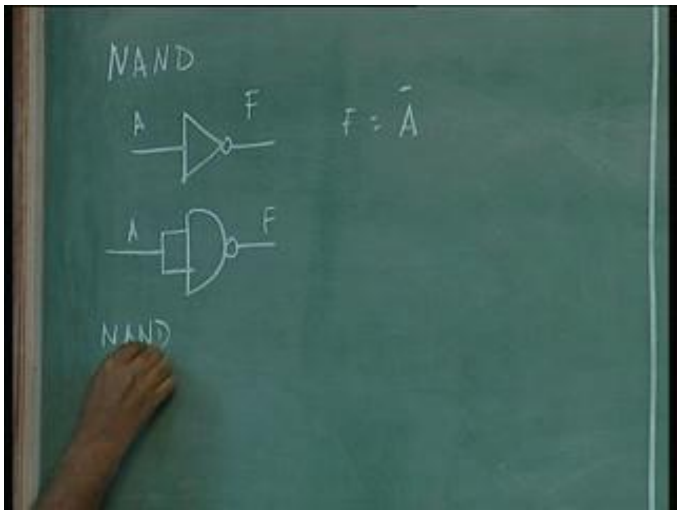
Combinational Circuits

In the last lecture we talked about the basic gates and we also the said the AND OR Invert combination can implement all logic functions. In addition to this AND OR Inverters we also saw few more gates like NAND, NOR, Exclusive OR, and Exclusive NOR or equivalence. I also mentioned in the last lecture that even though all functions can be implemented using AND, OR, Invert practically any digital system can be implemented using AND, OR, Invert.

Sometimes for reasons of practicality or the type of design or because of inventory available or because of technology reasons we go for other types of gates like NAND and NOR. And NAND and NOR have a special place in the sense they are called as universal gates, I already mentioned this to you. That means we can implement a whole logic whatever you can implement using AND, OR, Invert combination can be implement using NAND gates alone similarly NOR gates alone. Thus NAND and NOR for this reason are called universal gates. So I will quickly tell you why this is so.

Why is a NAND gate or NOR gate a universal gate. So let us look at a NAND combination. if you assume my premise that any logic function can be implemented using AND, OR, Invert that we will see later on more elaborately I don't want to prove that to you now in order to take my word for it at this point. So if you assume that then all that I have to show you is how to use a NAND gate to get AND gate, NOR gate and a NOT gate.

(Refer Slide Time 5:34)

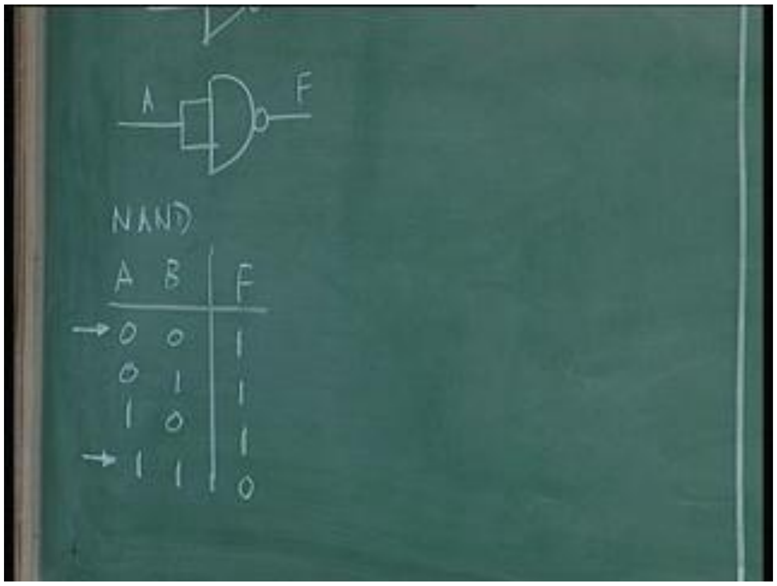


That means if you are going to believe me when I say you can get any logic function any digital system to be implemented using AND, OR Invert and if I prove that NAND can be used for each one of them then you know that NAND is a universal gate. That is the way I am going to prove because I don't want to be spending a lot of time on this. So we know how to get AND out of NAND because NAND with an inverter is an AND so first let us see how to get an invert using NAND.

What is an inverter if you remember from last lecture? F = A bar. Now let us have a NAND gate, if both inputs tied together and call this A, an F, now in the NAND gate if you remember the NAND gate truth table you remember I call this truth table (Refer Slide Time: 5:38) this is NAND gate.

Now I am making A and B the same. That means A is 0, B can be 0 or 1 because there is only one of them, F is 1. These are only two combinations possible; A and B both 0 0, A and B both 1 1. So if both are 0 0 output is 1 that means if A is 0 output is 1 if both are one output is 0.

(Refer Slide Time 6:42)

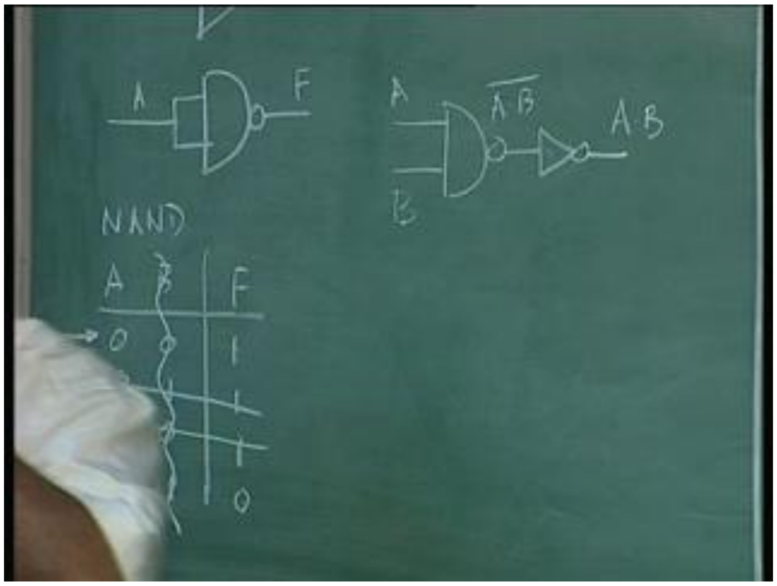


So a NAND gate with both inputs tied together is an inverter. That means I don't even need this and A is 0, B is one and this row is also not there. So, if A is 0 output is 1, A is 1 the output is 0 this is the combination so NAND is an inverter.

All you have to do is to get an inverter of NAND is tie to get the together and apply the input variable the output is the compliment of the input. and once you know that NAND can be used as an inverter NAND can be used as AND gate because NAND is nothing but AND inverse so invert it one more time, (Refer Slide Time: 7:33) this is A B NAND so this is NAND invert of this A and B, this is an invert this bar will be removed this bar is because of it is a NAND and this is the bubble.

I told you the bubble is the key thing bubble inverts and bubble also removes the inverts, one bubble inverts and the next bubble removes the inverts.

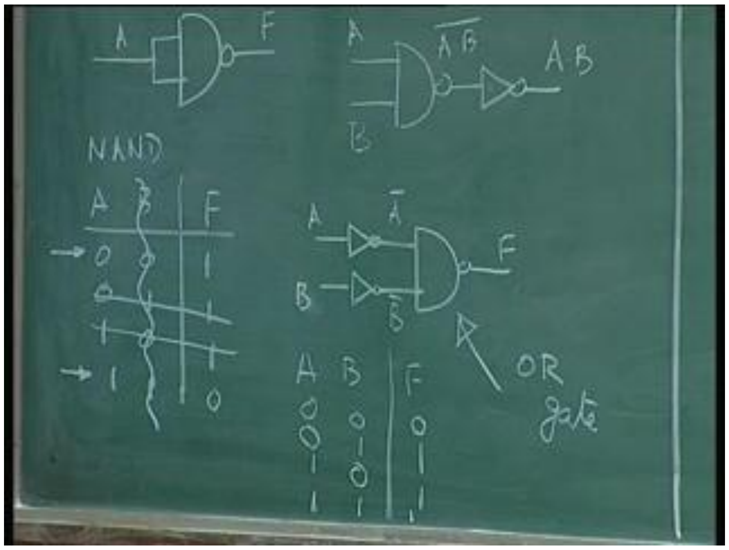
(Refer Slide Time 8:11)



So you have the bubble here with an invert, this bubble removes this invert sign so this is a AND gate. So I can get a NAND from AND by putting a NAND in series with inverter and inverter itself is to be got of a NAND by tying both the inputs together and applying the input.

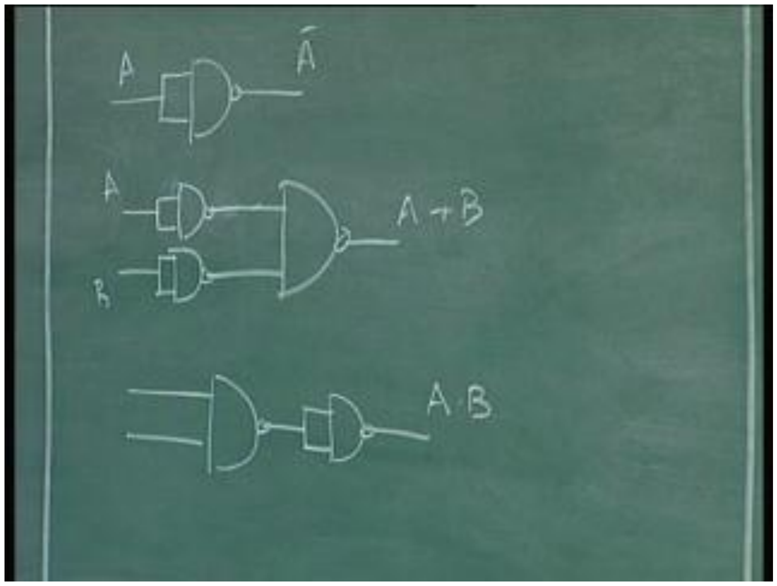
Then the third thing is of course the OR gate, how do you OR out of NAND? So let us take a NAND gate with two inputs A and B and invert them each. I tie an inverter to each one inputs of the NAND gate, I apply my variables to A and B instead of NAND inputs to the inverter inputs. That means this is here A prime or A bar, here this is B bar, there is a theorem called De Morgan’s theorem, you might not be heard of that we will see it later, without going to that now since I did not introduced it I can write a simple truth table A B and F.

(Refer Slide Time 10:26)



If A is 0 and B is 0 this is 1 (Refer Slide Time: 9:31) so we have 1 1 to the NAND gate and output is a 0, 0 1. When I put A = 0 and B = 1 this becomes 1 and this becomes 0, when I put a 1 0 then it is 1, 1 0. Put A = 1, B = 0 this is A bar which is 0 this is 1, 0 1 input output is 1, 1 1. When I gave A = 1, B = 1 this becomes 0 0 when I get 0 0 to the NAND gate the output is 1. That means this is the property of an OR gate, so this is an OR gate property. So I get a property of OR gate by taking a NAND gate and inverting each input. I get a property of AND gate by taking a NAND gate and putting an inverter at the output. And I get the inverter itself by taking a NAND gate and tying both the inputs together. So to summarize here then a NAND with two inputs tied together is an inverter, a NAND with inputs if you want to put it in the NAND way, this is what it is A OR B, this is A AND B.

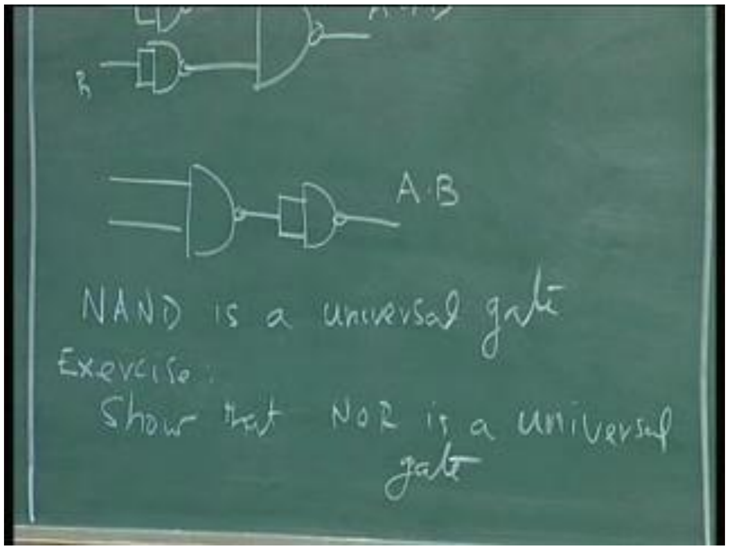
(Refer Slide Time 11:38)



That means I can use a NAND gate to get an inverter, I can use NAND gates to get AND gate, I can use NAND gates to get OR gate, I can use NAND gate to get an inverter and because of that NAND gate can replace every gate in an AND, OR, invert combination. And as I said if you accept my premise which I made in the beginning that any digital function even sequential which we will see later, any digital function any digital system can be implemented using AND OR invert and a NAND gate can alone do the job that is why NAND is a universal gate.

Then I can use only NAND gates think of this as a process by which I have to implement a given function or a given circuit of different complexities of a specific complexity where that should be some AND gates, some OR gates, some NAND gates. We are not aware of this stage circuit requirements or the technological requirement or process requirements as of now but surely it will be nice to have only uniform type of things to be done either circuitry-wise or technology-wise either for making something or designing something if everything is identical is it not easier can you sort of intuitively visualize that an all NAND solution may be a little simpler to do than a AND OR invert solution from the point of view of a circuit design? May be we don't know at this stage whether it is going to use more NAND gates or less NAND gates compared to the original number of gates.

(Refer Slide Time 15:45)



The number of gates we don't know at this point in time. it looks like we have to put more gates because we have to put two inverts for OR gate one invert for the AND gate and all that but later on we will show you techniques by which it is not all that extra gate there may be some extra gates but there will be some knocking up, there may be some invert.

For example, I may have an invert at the output of a function and an invert to the input of the next function and I can cancel those two inverts so two inverts I can save because of one invert I get another inverter so I can save them. Therefore some effective design techniques can be used to improve the density let us not worry about that in principle though it is better to make a circuit. More than from a designer's point of view from technology point of view from the point of making these ICs it is a very complicated procedure, a complex procedure it is nice to have identical things to do.

So, from that point of view people prefer this type of implementations. that is why I said even though all are called basic gates AND, OR, Invert, NAND, NOR, Exclusive OR, Exclusive NOR these three are fundamental gates AND, OR, invert because in a mathematical sense or a logical sense AND, OR, invert are required to implement any given digital function or digital circuit digital system but it will also be equally well done using NAND or NOR that is why NAND or NOR are universal gates.

Now I am not going to do the same thing for NOR do the same thing for NOR.

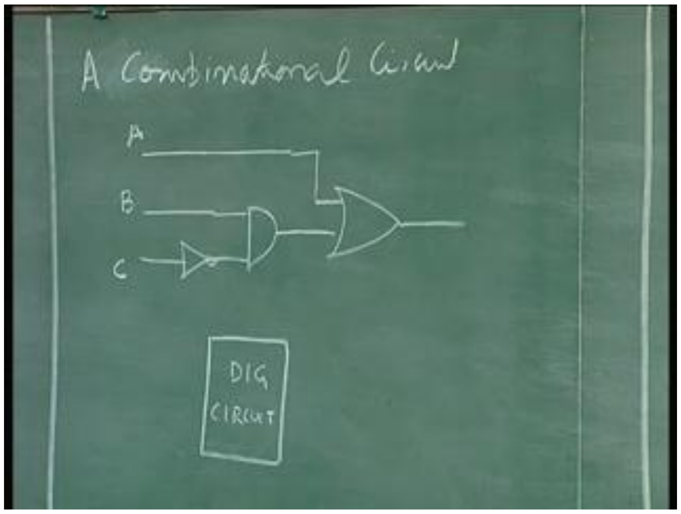
Exercise:

Show that NOR is a universal gate, very simple. Same way you did this, you start with the inverter and OR gate and NAND gate as to how to get each one of them using a NOR gate or using NOR gates. Then in that case I have now a variety I have a choice of making a design. When I want to build a circuit I have a choice of using this or that or combination their after or whatever. That gives me more flexibility in terms of design in terms of the technology I can choose and things like that.

We have introduced you to the concept of digital systems, a digital signal and two levels binary and how binary alone cannot do the job because we cannot capture all the variations of a signal by using only two levels so you have multiple signals or multiple bits and multi-bit signals which can capture the variations, nuances of the variations fineness of the variation and then we said the thing is to build all these functional blocks which can do all the things we need to do and we said that there are two types of circuits combinational and sequential and we said we will start with the combinational and for the combinational the basic building blocks are the gates so we introduced gates. So this point in time what we will do is use these gates to implement a few circuits. We will see how to get a circuit using gates for a given specs that is the design or given a circuit how do you analyze to get the function.

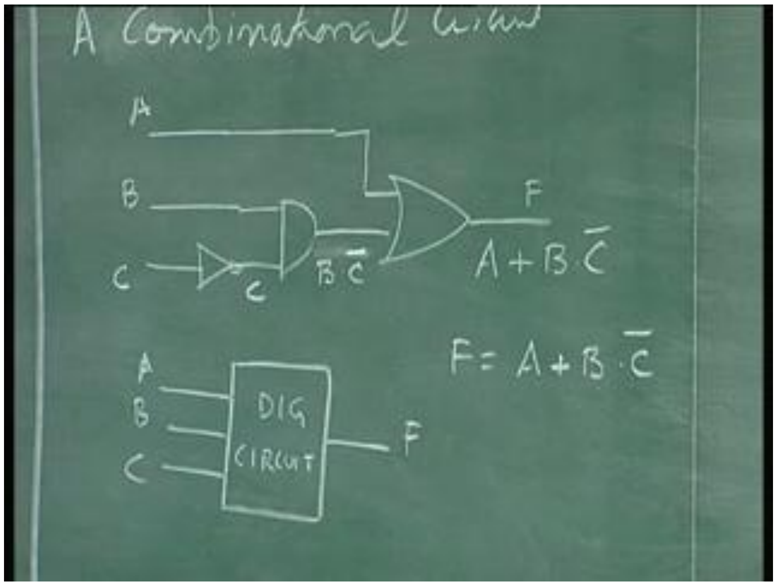
So let us take a simple example of a combinational circuit. Since I said most circuits are implemented using AND, OR, all circuits in theory can be implemented using AND OR Invert I have taken a simple circuit in which AND OR Invert are all used. Let us call this signal A B and C three variables so I have a digital system or circuit or a subsystem. Don’t get confused too much about this definition what is called a circuit, what is called a subsystem, what is called a system as it is all sort of a definition level, anything that we want to do we do. If it is small we call it circuit, if it is slightly more complex we call it subsystem, huge we call it a system like a micro processor we call it like that, nobody will say micro processor is a circuit even though it is a circuit. The components are connected and it behaves the way we design so it’s a circuit if that is the definition of a circuit.

(Refer Slide Time 19:21)



At the same time a micro processor is too huge so you call it a system you would like call it a system a computer system and a smaller thing we call it, adder, subtracter, multiplier, arithmetic functions of a micro processor we call them arithmetic subsystem, arithmetic sub-unit or arithmetic sub-module or if it is a simple gate structure like this I would like to call it a circuit because of couple of gates. So I don't think you should worry too much about the definition or the use of the word in fact I use them interchangeably all the time circuit or a subsystem. I won't unnecessarily use subsystem and all that. When you are talking of a system and you break it down into different modules and try to implement each one of them separately at that time only I will leave subsystem. But sometimes system and circuits are sort of interchangeably used, don't get too much agitated over this. So this digital circuit has three inputs A, B and C and one output F. since all of us know the functions of the gates now this is B, this is C this is C bar (Refer Slide Time: 20:07) so this is B AND C bar and this A OR B AND C bar that is behavior of the circuit.

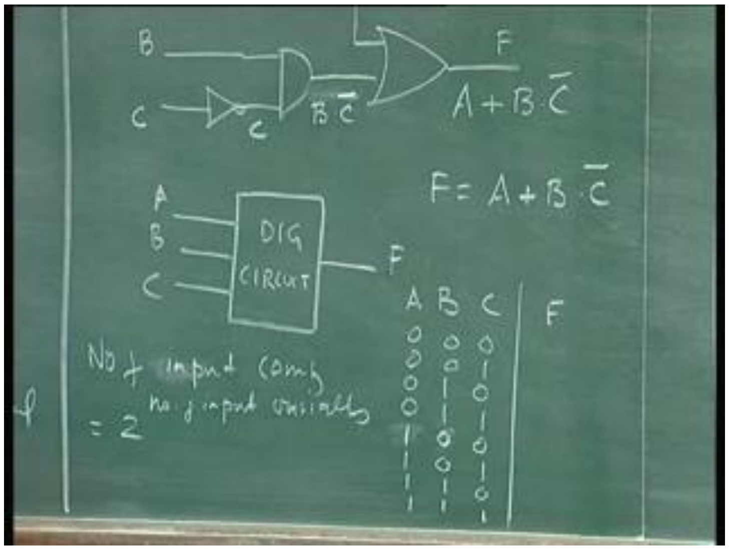
(Refer Slide Time 21:06)



So the functional description of the circuit is F = A OR B AND C bar but if you want to be very correct about it say A OR B AND NOT C where F = A OR B AND NOT C. This is the functional description, this is the circuit description and then the third one is the truth table. The operation of course involves AND operation, NOR operation, NOT operation. When I introduced the gates first I said that an operation is defined by gate, a symbol is defined for a gate, a functional description of the gate is given and a truth table is given. in this case the operations are here, this is the symbolic representation and the circuit representation (Refer Slide Time: 21:36) this is the functional representation so the last thing is the truth table where I will have A B C as inputs and F as the output so we will quickly write all the possibilities of the inputs.

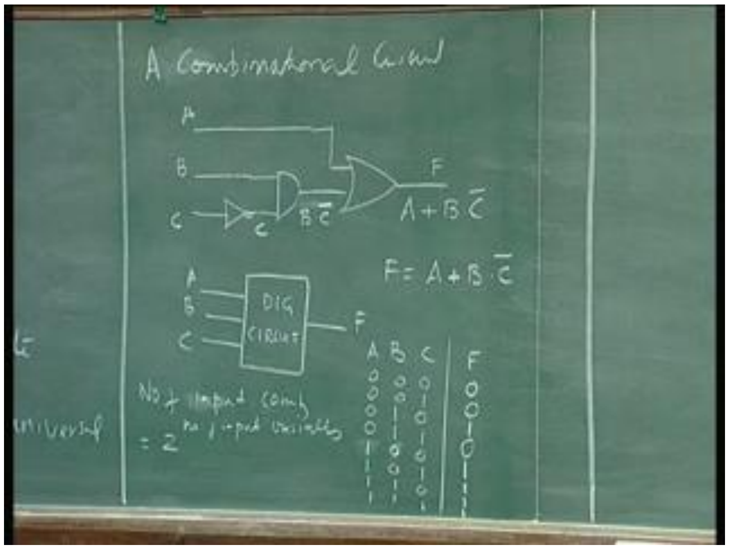
How do you know all the possibilities? There are three input variables, there can be eight variations possible because each input can be either 0 or 1 or true or false symbolically z 0 1 so each of the inputs can be 0 or 1 there are three inputs so there can be combinations of each one being true or false so there are eight different combinations. How do you know that 8 is 2 power 3? The number of combinations is 2 power the number of input variables. So it is 2 power n number of input combinations is 2 power of number of input variables. It is also the same as the number of rows in the table 0 0 0, 0 0 1, 0 etc.

(Refer Slide Time23:25)



Hence there are eight rows in the truth table because there are eight combinations and you should see what the output is for each of these combinations. Because you know the operations you can quickly do that. When A = 0 B = 0 this is 0 1, the moment this is 0, this is 0, this is 0 so the output is 0. This is 0 again, this is 0 again, this has to be 0. So if I make a mistake please tell me don't be watching and then finally say Sir it should have been that way. Then when A is 0 B is one that means this will be 1, C is 0 that means this is 1 that means there is 1 here and output is 1.

(Refer Slide Time 24:46)



It is 0 1 1, this is going to be 0 because the one here is going to be translated to 0 here and once this is 0 this is going to be 0 since it is 0 the output is 0 and for the rest of the combinations I can close my eyes and write one because A is 1. It is an OR gate so for an OR gate input one of the inputs is 1 so output is 1 for all the four combinations of the last four combinations for last four rows the input is 1 A is 1 so output has to be 1. That means output F is depending on the input variables A B C can be defined as this A OR B AND NOT C (Refer Slide Time: 24:56) and this is the output so the whole thing is called the truth table. So I have the circuit logical representation or the functional description and truth table.

What does it mean? There is some meaning. we cannot just design a circuit arbitrarily of three variables and make those variable change and then say output is 1 or output is false, output is false when A is false B is false C is false, output is false when A is false B is false and C is true and so forth, output is true when A is false B is true and C is false and all these we can keep on saying. It should have a meaning; it should have a meaning in terms of the control function the circuit function.

For example, if we have an amplifier I told you in the beginning an analog device analog circuits we have an amplifier, I have a sound signal converted into an electrical signal by a transducer of the microphone then gets amplified and the amplified signal is an electrical signal which is converted back into sound signal by a loud speaker.

Now what is the function of this circuit? You can say it is an amplification function because my voice cannot be heard throughout the room we have an amplifier to increase the voice, you can ask me to shout the other alternative is to put an amplifier. Like that what is the meaning of this circuit? That is what the meaning is; you should have a meaning in life right? This circuit has to exist to with a meaning in its life otherwise there is no point in existence. It can be anything, for example, take a simple device, this may be an event occurring, this could be an event occurring in a controller, bigger system like a micro processor when does it get reset as a simple example.

You get an error message, when does an error message come? On certain conditions being fulfilled an error message comes. Those conditions are represented as variables and when that combination of conditions occur then it happens or it can be a chemical plant, there has to be a shut down if there is a set of parameters taking certain values. So the chemical plant or it could be a nuclear reactor then you will say nuclear reactor becomes more dramatic it has to be shut off if certain things happen and how do you monitor that? Somebody has to put a control circuit that control circuit will have variables called A B C and that A B C variable takes these particular values, the value itself may not be important but the combination of the values may be important. When the combination of the values A B C occur in this way then automatically the circuit should trip so that you are save, some systems have such thing. Therefore like that if you want to give a physical meaning to that we can think of something like this.

Supposing you want to know, let us take a common example from everyday life. Supposing you want to go home let us say A is a Sunday, you want to go home you normally go home on a Sunday let us say or you go home on a Sunday. You also want to go home whenever it is possible other than Sunday also. Let us say you are one of those gem of a student who doesn't want to miss a class, of course you are not that I know.

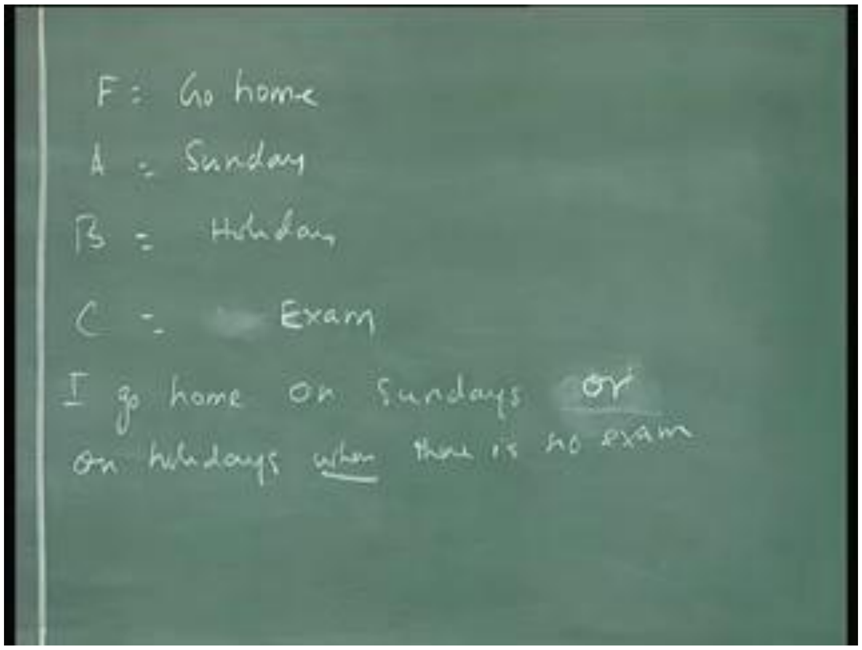
But let us for a moment assume make believe assume that you are all very sincere students you don't want to miss a single class of mine you are so interested in my class so I will say I will not go unless it is a holiday. So let us say B is a holiday so I will go home if it is a Sunday or if it is a holiday then A may be a Sunday, B may be a holiday other than Sunday but I don't want to go on all holidays but you are so sincere that even it is a holiday you don't want to go home if there is an assignment due the next day or there is a quiz next day, exam next day. So C may be the quiz so if there is no quiz and if it is a holiday I will go home or if it is a Sunday I will go home.

This is some sort of a simple explanation because in a real life situation I can explain like this. of course in a chemical plant you can imagine a water level or a chemical level or a fluid level, fluid flow or temperature raising or any aircraft landing system supposing the pilot sits in the cockpit and finds so many values and some of the values are so critical for landing or taking off if he does not reach those values he doesn't want to take the plane or if he is flying he wants to land immediately if certain combinations occur. These are done by monitoring these values levels or transducers or various instruments and when those things happen you will land, automatically the landing lights will be on and whatever is the landing procedure will take over, if it is an autopilot mode it will go to the autopilot mode so that the pilot will let the plane land by itself and all those things.

Or it can be a simple welding machine example. Supposing you want this and you should have cash to put in that and the beverage you want is available to you so these are the different conditions. Just because I want a coffee it does not mean I can go and pray for coffee in front of the coffee machine. So I will have to make sure that it works. That means I will have to put the correct coin and then supposing some machines give you change some machines do not give you change so I have to press the right button. So there should be power coffee must be available I should put the correct combination then the particular coffee or beverage will be dispensed. So I can think of and imagine many situations whether you can do much more than that, you youngsters are supposed to be much more imaginative than us so you can think of hundreds and hundreds of such situations where this type of control is required which are implemented in terms of digital circuits and put in place so that finally the whole system works and you go without any worry and you sit down in the plane and then relax and then the plane lands in the destination and then you go home. But many things happen inside the circuit.

Therefore in this case I am going to then say F is go home, A is a Sunday, B is a holiday, C is exam. So I go home on Sundays or on holidays when there is no exam, when there is no exam the next day I don't have to say all that. How can you have an exam on a holiday don't ask me all those questions. I am trying to give you an example to understand but you can think of this as parameters critical in a chemical plant or aircraft landing or washing machine or an elevator, you don't want to get stuck in an elevator that is worst than a plane crash at least there you will die here you don't know what is going to happen to you.

(Refer Slide Time 33:24)



F : Go home

A = Sunday

B = Holiday

C = Exam

I go home on Sundays or on holidays when there is no exam

These are the things you need to know. Now, given a circuit I should be able to analyze it. Draw the truth table and draw the inference and finally say whether this is what you really wanted. Is this what you really wanted? Is there any combination for which it gives an output when it is not required or should not be given that's called a glitch, it is an error. Sometimes I get an output, an error on the safe side is no problem. Suppose the plane does not take off when it is supposed to take off, okay you will curse the pilot or the airlines but nothing would happen. But the other thing should not happen when it is not supposed to be flying it should not be flying right, so there should be no error. That means what is the purpose of analysis of a circuit? make sure that outputs are required at the right time given in the right time no output is available when it is not supposed to be available, or output is available at all the times it is supposed to be available both are important. So I should have a 0 exactly when I want the 0 as the output. I might have 1 whenever 1 at the output, this is very important. I took an arbitrary example here.

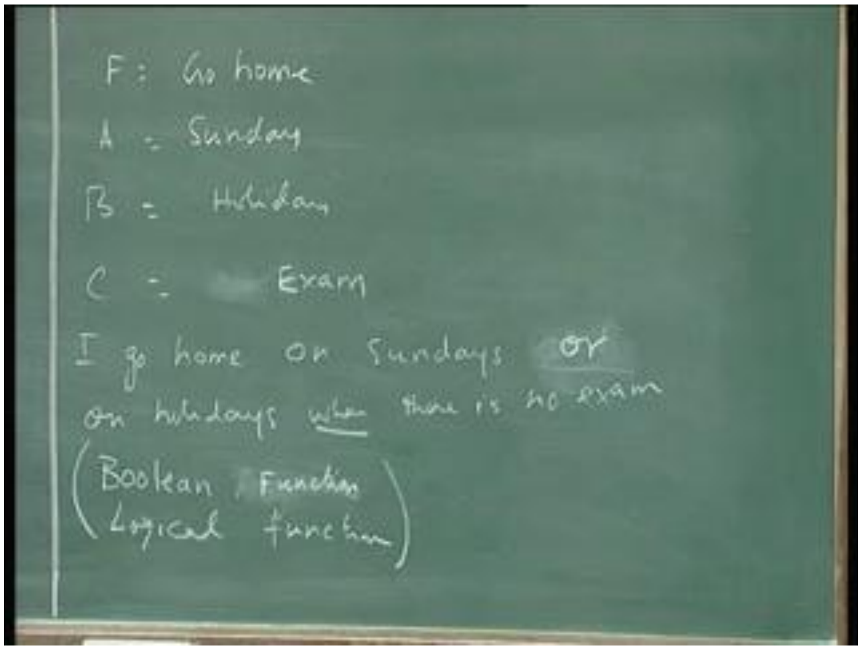
Now the question would be; this circuit somebody gave me I analyzed it and I said that I am trying to logically prove that is something that is going to do its job but is it the only way to do this? Is there a simpler way of doing this? How do you know whether this circuit is the simplest possible circuit to do the same operation? Now forget about holidays, going home and exams. A B C are three variables. I want an output high or true whenever A is present or B is present and C not present is it not, is it not, what it is? Output should be true if A is true or if B is true and C not true, is it the only circuit possible? We don't know how we got this circuit disorder because I gave it to you; you wrote the truth table and analyzed it. So this may not be the best possible way to get the circuit. I don't know we have to simply it, we have to see it, we have to look at the possibilities but this circuit can be simplified and still got in the same function. So we need mathematical tools to find out whether a given function, suppose the problem be given to you like this is it not?

This is the reverse thing I told you but really it will be given like that. I would like to go home on a Sunday whenever is a Sunday or a holiday when there is no exam then you will program it let us say just imagine you program it in a computer and then it remains 0. Tomorrow is a day off so I have to go home or it is an alarm. In the case of A B C being variables when A is true or B is true and C is not true it is supposed to bring an alarm or light a bulb then I go and fix a circuit and then get these A B C and then power it up, connect power supplies, connect the lamp at the output which will glow and whenever the lamp glows I know that the system has been switched off or it is automatically switched off.

Now, having given this specification how did you arrive at this? It is because you simply took this and directly implemented it. I took the Boolean function this is called the logical function or Boolean functions, these binary digital functions are called Boolean functions because the whole thing with the mathematical background was created by a person named Boolean so because of that this is called Boolean. Therefore it is a Boolean function we will call it or sometimes they say logical function and given this I can directly interpret in the circuit it is easy. But this is a very simple example of three gates in fact including inverter. It is not expensive I can use three gates.

But suppose if I am thinking of a system or a subsystem with one thousand gates some complex example of an aircraft landing I told you aircraft landing is not going to be as this simple is it not? Aircraft landing is going to be like several parameters to be monitored and processed, it is not that wind pressure will be directly used as a transducer wind pressure will be coupled with speed, knots and all that, altitude and temperature inside will turn out and one equation will come. Hence there will be so many functions so the complexity of the circuit will be very large and that time it may be worthwhile to think whether directly given the equation should we implement it or see whether the equation can be simplified further and then implemented.

(Refer Slide Time 37:37)



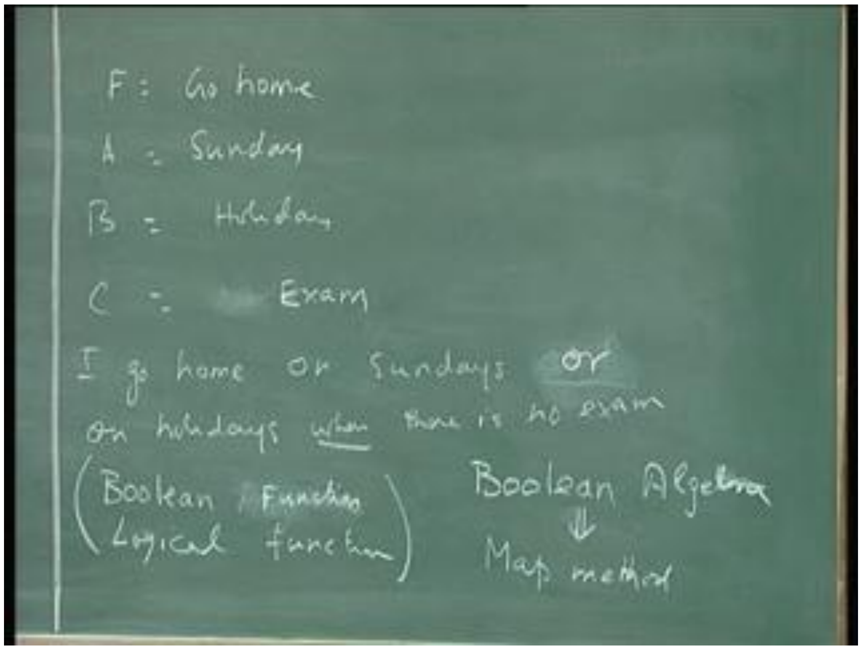
When you simplify the equation further and implement it you save on hardware. when you save on hardware remember the mantra I told you when you save on hardware it increases the performances in terms of whatever performances it is such as speed and so on, cost is lower, reliability is more because fewer connections interfaces inter-connections, power dissipation is lower, cost is less, size is small everything. Remember that as a digital mantra namely high performance, low cost, small size, low power, if you want to add reliability being highly reliable, more and more components are going to be counter productive in all of these factors. So it should have the same function performed with less and less of those components. There are two approaches; one approach is to going in for components which are not at gate level but at a slightly higher level of modules performance-wise, that we will see later. but before we do that in a gate sense it is only using the AND OR Invert or NAND NOR that type of gate level implementation so there are lots of things you can do to improve or to reduce the complexity of the circuit. The given function need not be directly on to a circuit. Given a Boolean function or a word description, this is a word description, (Refer Slide Time: 40:12) this is the Boolean description.

Given a Boolean function or a word description it is not necessary to jump to the nearest gate and then start connecting it. You always examine it and see whether it can be simplified further. There are tools available for it to make it systematically. One such tool is called Boolean algebra.

As I told you the whole thing is a digital binary, binary variables, things with binary variable, binary functions using logical operations like AND OR Invert whole thing is the name, it is named after Boolean, Boole is the name of the mathematician so these are Boolean functions, Boolean variables. Similarly the algebra or the mathematics that is used to simplify the Boolean function to a equivalent function which is performance-wise identical but use as less hardware less number of gates that tool is called Boolean algebra. So we had to learn Boolean algebra. I am not going to teach that, first of all it is not a great thing, it is a very simple set of rules. There are only binary variables; a variable can only take two values 0 or 1 true or false. And even the Boolean algebra sometimes, whenever there is mathematics involved you have to apply it properly so you will never know whether you have applied it in the right way. you always have a doubt at the end of the day after applying a Boolean algebra to a given Boolean function you got a simplified function which is identical to the first function, you are not sure whether this is the simplest or can it be simplified further, who is going to tell you that.

I give you a problem in the exam; given a Boolean function simplify it to as few variables as possible using as few operators as possible. There are two things you have to be worried about. One is the number of variables and the other is the number of operations. So you will have a few operators as possible, operations as possible and as few variables as possible. So given a thing and then reduce it to the minimum possible number of operations minimum of variables I can take. I gave a function with eight operations and nine variables and somebody got three variables and two operations, somebody got five variables four operations we don't know who is right who is wrong which is the best fit.

(Refer Slide Time 42:58)


F : Go home

A = Sunday

B = Holiday

C = Exam

I go home on Sundays or on holidays <u>when</u> there is no exam

(Boolean Function / Logical function)

Boolean Algebra $\Downarrow$ Map method

This is the algebraic method the other is called the map method. From here we will go to map method where you can do a graphical simplification of getting the simplest possible solution for the given Boolean function, these two things we will see. Boolean algebra we will see quickly. I am not going to spend much time on that for many reasons, one is that as I said the map method gives you a much better easier way of getting the same goal, we can get something more easier and more reliably. Map function always gives you the best possible result, minimum possible solution in terms of the number of gates and number of variables.

Number of variables and number of operations I get the map method always reliably the lowest. When that is the case why should I got to another method which is not so reliable unless I apply it carefully, unless I exhaust all possibilities I am not sure what I am going to get. The second thing is gate saving is no more a big issue today. because gate saving was supposed to be a very important in the early days of all this hardware being developed and it's all very expensive to go through, it is something like inflation in terms of electronics. Probably when you talk about earlier days when you want to save a rupee today you want to save hundred rupees, it is not worth saving anything less than hundred rupees, you keep the change you can say all of you especially you are spending your parents money you will be even more generous.

What I am trying to say is the technology is advanced so much in a small area of silicon they can put thousands and even millions of gates so what is the meaning of saving thousand of those million gates. If you have one million rupees if I ask you thousand rupees of course I don't know you will give me but I am just saying we won't mind it know at least somebody to whom you want to give you won't mind giving may be not to me but to somebody. But if you have only two thousand rupees when somebody asks thousand rupees you will think twice. So the saving of the gate concept came at that time and today gate saving is not a big issue because now we go for system level and subsystem levels where we are interested in functional minimization, functional efficiency all those parameters I told you, again I repeat power, reliability, cost, etc and when that is the issue we will not worry about having one extra AND gate there and one extra OR gate there.

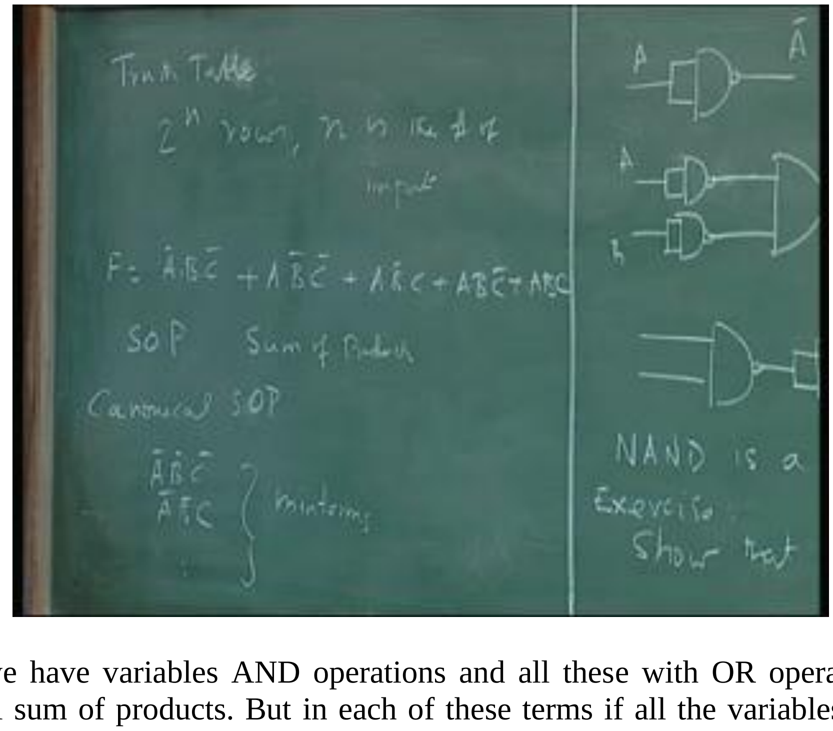
As I have gone through this course for completeness sake I will quickly introduce you to Boolean algebra may be in half a class or so and we will spend a little more time on map method because it is a systematic graphical method of simplifying a given Boolean function to get the possible minimum possible hardware solution for a given function or a given description of the problem. We will see that in subsequent classes but before that I want to give you some definitions today before finishing it off.

We said that this is the truth table of the system. We already defined truth table it has all the input combinations in the corresponding outputs. I will not write the definitions here truth table will have all the inputs and the corresponding outputs, all the possible inputs combinations and the corresponding outputs and the number of entries in the truth table we know that it is 2 power n where n is number of inputs. So number of rows 2 power n rows where n is the number of inputs.

Now each of this is called a min term. for example instead of saying this f is true if A is true or when B is true and C not true that is one way of saying if I did not know this function and if you did not give me this word description of the problem or if you did not give this Boolean algebra or if you did not even give me this circuit and you only gave me this truth table I will read this as F is true when A is false B is true C is false, F is true when A is true B is false C is false like that I will start reading. So you can write this F as from your truth table directly as in this case as A bar B C bar OR A B bar C bar A B bar C A B C bar A B C. In each of these outputs this is called the sum of products expression SOP, it looks like a sum plus, and even though it is not a plus it is an OR operation since we are used to arithmetic and each one of these is an AND operation A bar AND AND but it looks like a product, it looks like A times B times C so this is called a sum of products. The output can be expressed as a sum of products directly from the truth table. Of course it can be simplified later on I told you by either graphical method or Boolean algebra method. When somebody says sum of products this should be like this and this is particularly called canonical sum of product that means in canonical sum of products each product term will have all the variables A B C either in the true form or the complimentary form. That means from the truth table if you read the sum of products it will be a canonical sum of product.

When you simplify the canonical sum of product you will get a minimum sum of products so you will have to first write the truth table get the canonical sum of products either simplify it in a Boolean algebra way or map method and then get the minimum sum of products and each of this term is called a min term; A bar B bar C bar, A bar B bar C, etc are called min terms. So these are the definitions. This is truth table (Refer Slide Time: 49:43) the function is expressed as a sum of products.

(Refer Slide Time 50:02)



Normally we have variables AND operations and all these with OR operations that is why it is all sum of products. But in each of these terms if all the variables are present either in the true form or the compliment form it is called canonical sum of products. That is what you can directly obtain from the truth table and simplify to get the minimum sum of products. Or it need not be minimum you can simplify to get a sum of product if you simplify it long enough you will get the minimum sum of product and that can be implemented using hardware. So I want to know this truth table, I want to know the sum of products, what is meant by a product term, what is by a min term, min term is what we have in the truth table directly and the term canonical refers to the output directly read from the truth table.

So with these definitions next class I will briefly give you an overview of Boolean algebra very quickly and then we will go on to the map method of reducing the given Boolean functions so that we can get the minimum sum of products from which you can get the most efficient hardware for a given function.