

Computer Organization and Architecture: A Pedagogical Aspect

Prof. Jatindra Kr. Deka

Dr. Santhosh Biswas

Dr. Arnab Sarkar

Department of Computer Science & Engineering

Indian Institute of Technology, Guwahati

Fundamental of Digital Computer

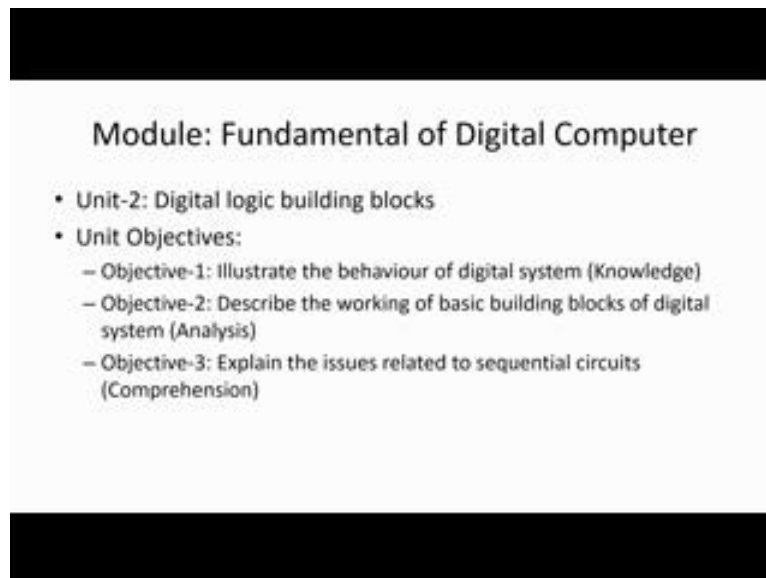
Lecture - 02

Digital Logic Building Blocks

Hello everybody welcome back to the online course on Computer Organisation and Architecture. So, we are in the first module, first module is based on your fundamental of digital computer and we are in the unit 2, Digital Logic Building Blocks. Now, we are going to see what are the things that we are going to cover in this particular unit.

So, as we mentioned that we are going to first mention about the objective. Let's see what are the objective that we have today.

(Refer Slide Time: 01:00)



The slide is titled "Module: Fundamental of Digital Computer". It lists the following content:

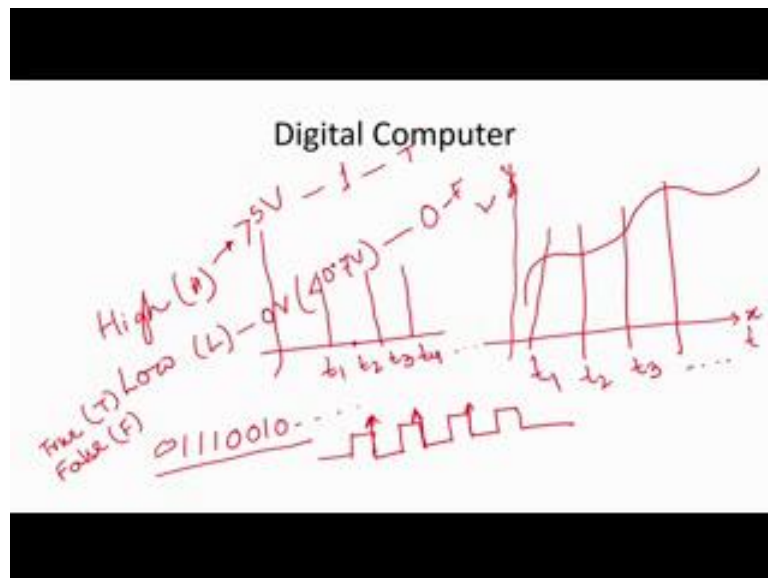
- Unit-2: Digital logic building blocks
- Unit Objectives:
 - Objective-1: Illustrate the behaviour of digital system (Knowledge)
 - Objective-2: Describe the working of basic building blocks of digital system (Analysis)
 - Objective-3: Explain the issues related to sequential circuits (Comprehension)

So, in this particular case we have mentioned 3 objective, objective 1 illustrate the behaviour of digital system. Again we are going to cover this things in knowledge level because these are the things that we will be needing while going to discuss about computer organisation and architecture. But if you look into the objective so if you are going to look for the behaviour of digital system in another course, say the digital logic design in that case these objective may

go into either analysis level or maybe in the design level. But here we are just going to touch up on it and we are going to give the information in knowledge level and we will be using this knowledge while discussing about the fundamentals of computer organisation and architecture. Objective 2, describe the working of basic building blocks of digital system. So, here we will slightly give ideas about the analysis of the digital system. And objective 3 explain the issues related to sequential circuit, ok we will come back to this point what is sequential circuit.

Now, we are talking about digital computer or this modules is fundamental of digital computer. When we talk about digital computers necessarily another question will come to our mind that is there some other computers are there. So, in that case you can look into this issue and you can say that yes you can categorise the computer in two dis two broad categories, one is a digital computer and second one is analog computer here in this particular course we are going we are mainly going to emphasis on digital computer, what is a digital computer and what is analog computer. So, this computers are electronic devices. So, it works on electrical signals that signals may be either voltage or current. So, in case of analog computer we work with continuous signal.

(Refer Slide Time: 03:01)



So, say that if I am going to draw a behaviour say some time and in this particular case say you are having the voltage speed in x-axis we are having time and y-axis say we are having voltage. So, this electrical signals flows continuously then we say this is your analog signal. So, in case of analog computer we are going to work with continuously varying signals, but in case of

digital computer we are going to sample these things that signals in particular instance of time and we are going to look for those signal values only. Say for example I can have some timing instance I can say what is the signal value of time t_1 , what is the signal value of time t_2 , t_3 etcetera. So, in this way we are going to look into it; that means, we are going to work with discrete signal. So, digital means discrete signals.

So, when we are going to talk about digital computer you are mainly you are going to concentrate on two level of signals either at some time instance the signal is high or in some instances signal is low something like that. So, in t_1 signal is high, t_2 signal is low, t_3 signal is high, again t_4 signal is high like that. So, for sampling this thing basically we work with a clock signal. So, basically we are having a continuous running clock and we are going to sample the signals at some instance of time. So, maybe somewhere here.

Ok and secondly, I am saying that you are going to work with discrete signals and we are going to see the voltage values at some instance of time. So, this value voltage value may be either high or low. Ok so, in some cases we represent it by H or low. And generally when talk about high voltage basically it is depending on the technology it may be something voltage more than 5 volt in some cases it may be 8 volt in some cases it may be 12 volts and for low means it is basically 0 volt or maybe a very low voltage which is less than say some 0.7 volt or something like that.

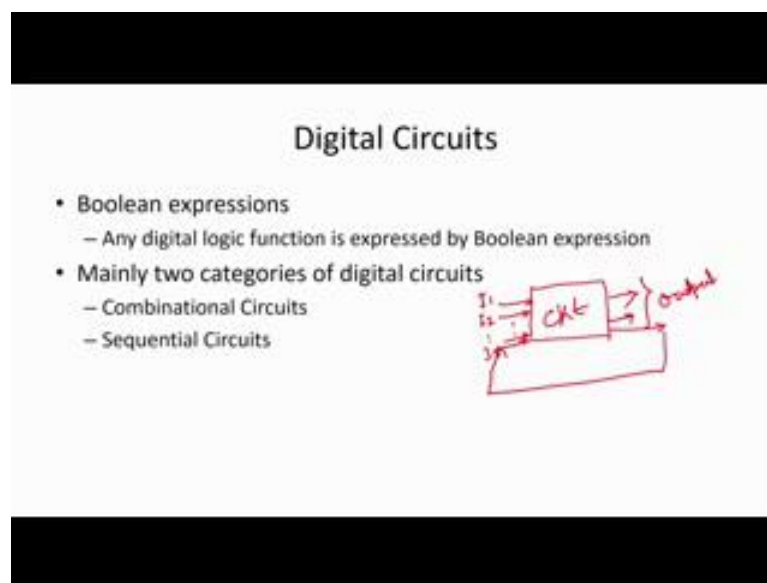
So, if voltage level falls below this particular level generally we say that this is your low signals and when voltage level goes above some certain value we say that this is high level. So, generally we represent these two steps as high and low. For our convenience that high may be represented by 1 and low will be represented by 0. So, basically it is a combinations of 0s and 1s only at some instance of time the signal value is high that is represented by 1 and in some cases the signal value is low in that cases it is represented by 0. So, eventually we are getting some signals value which is you are say 0 1 1 1 0 0 1 0 something like that.

So, finally, what will happen? We are coming down to some number system which is your binary number system. So, in our discussion mainly we are going to concentrate or going to discuss with respect to this binary number system. So, digital computer works on binary number system. In top level we can say like that, but in low level we will say that it will work on either some high voltage or some low voltage. So, this is the way we can look upon the digital computer.

Again this digital computer is basically related to digital logic. So, in case of logic generally we are having 2 two values, one is called true and second one is called false. So, true is represented by t and false is represented by false. So, in some places what will happen 1 will be represented by your true and 0 will be represented by false. So, eventually all are going to have the same meaning. So, in case of digital computers we are going to work with discrete signal values of discrete signal may be high or low, high values will be represented by 1 or true and low values will be represented by 0 or false.

Now, what is the digital circuit? So, in case of digital circuit we can say that any digital logic function is represented by Boolean expression. So, if we write some Boolean expression that Boolean expression is going to evaluate some function and that function can be implemented with the help of your digital logic circuitry.

(Refer Slide Time: 07:41)



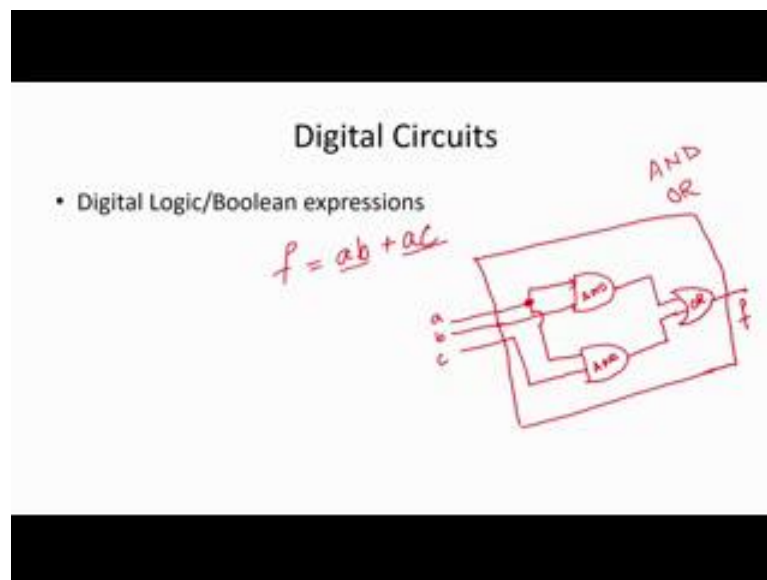
Again the digital logic circuit can be categorized into two different categories one is your combinational circuit and second one is your sequential circuit.

So, in this particular case I will just simply say that I am going to take a digital circuit I am going to just think it as a black box I don't know or you are least bothered about what is inside that particular box. So, here we are going to give some input signal say I_1 , I_2 to say I_n and we are going to get some output signals. Ok this output signals depends on the input signals and depending on the behaviour of the circuit we are going to get the values of those particular

output signals. So, such type of circuit is known as your combinational circuit. So that means the output entirely depends on my input this is a combinational circuit.

But when we talk about the sequential circuit then what will happen some of the output will be given as a feedback and connected as an input to the circuit; that means, the current output of the circuit depends on my previous output that previous output is coming as an input to my circuit. So, in that particular case we are going to say this is the sequential circuit and output depends on the previous output also. So, we have to have some mechanism to retain this particular previous output and we will see that particular things when you are going to talk about sequential circuit.

(Refer Slide Time: 09:32)



Now, there is a correlation between Boolean expression and digital logic. So, say that I am simply talking about simple functions say $ab + ac$. So, this is a Boolean function and a , b and c are Boolean variables; that means, it can take values either true or false, on other hand I can said it true can be represented by 1 and false will be represented by 0. So, if we are having such type of Boolean expression for every Boolean expression we are having a digital logic circuit. So, how we can implement? I can say that this is a AND b it says that values of these particular component is 1 provided both a and b are true, both a and b are 1 then the values is 1. Again this function is having two components ab and ac this is the OR combination or plus combination it says that the function value will be 1 either ab is 1 or ac is 1, so that means we

are going to talk about two operation one is your AND, and second one is your OR. For such type of operation we are having digital logic gates.

So, in this particular case what I can say that, I can say that I am having an AND combination two AND combination and I am having 3 input a, b, c . So, it is basically a and b is ending together and a and c is ending here. So, when both a and b and a and c are true then what will happen, the output of these two gates are going to be high or true. Now, main final function is the OR of these 2 combination, so we are going to connect one OR gate over here and finally, this is the functional value f .

So, these are the AND gates, two AND gates and one OR gates. So, this is a circuit and this circuit implements this particular Boolean expression. Now, we can say that now, I already mention that I can feel these particulars internal details as a black box and whenever I am going to give the input over here, depending on the input we are going to get the output. So, this is the way. Now we will see what are the different types of logic gates we have.

(Refer Slide Time: 12:01)

Combinational Circuits

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\neg	ab	$\neg ab$	$a + b$	$\neg(a + b)$	$a \oplus b$	$\neg(a \oplus b)$																																																																																																
Symbol																																																																																																							
Truth Table	<table border="1"><thead><tr><th>a</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></tbody></table>	a	x	0	1	1	0	<table border="1"><thead><tr><th>a</th><th>b</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	a	b	x	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"><thead><tr><th>a</th><th>b</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></tbody></table>	a	b	x	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1"><thead><tr><th>a</th><th>b</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></tbody></table>	a	b	x	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"><thead><tr><th>a</th><th>b</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	a	b	x	0	0	1	0	1	0	1	0	0	1	1	1	<table border="1"><thead><tr><th>a</th><th>b</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></tbody></table>	a	b	x	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"><thead><tr><th>a</th><th>b</th><th>x</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	a	b	x	0	0	1	0	1	0	1	0	0	1	1	1
a	x																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
a	b	x																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
a	b	x																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
a	b	x																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
a	b	x																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
a	b	x																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
a	b	x																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

$(ab) \cdot c = a(bc)$

So, logic gate basically we are having some standard gates and here in this particular course we are going to see in the gate level only, how those particular gates are implemented we are not going to look for it. That implementation issues will be discussed in some other courses may be in your digital circuit design. So, here first gate is NOT gate. What is that NOT? It says that if my input is high then output is low and output is input is low output is high. So, a is your input and x is your output. So, when a is 0 x is 1 when a is 1 x is 0.

Second gate is your AND gate. So, in AND gate already I mentioned that if the both the inputs are high then output is high. So, this behaviour is represented with the help of a table operation and these table is known as your truth table.

So, basically the behaviour of digital circuit can be represented with the help of truth table. So, here I am having 2 input a and b . So, when both the inputs are high then output is high otherwise output is 0. So, you just see this combination when both are high then output is high in other cases output is 0.

Similarly, we are having a NAND gate. NAND gate is nothing, but AND OR invert. So, first we are going to ANDing the operation this is AND then these bubble is going talk about the invert it. So, you are going to say this is the NAND gate. So, since NAND is the invert of your AND. So, you just see the truth table the output is exactly inverse of this particular output of AND gate, when it is 0 then output is 1 and when output of AND gate is 1 then output of NAND gate is 0. Like that we are having OR gate, in case of OR gate either a or b if any one of the signal is high then output is high otherwise it is 0. So, you just see that if both the input has 0 then output is 0, but if any one of this input is high then output is high. Similarly we are going to get a NOR gate, NOR is nothing but OR and invert. So, first we having the OR combination than this bubble again represent the invert, so you just see that or is inverted over here, when output of OR is 1 then output of NOR is 0. So, NOR is invert of this.

Like that we are going to have another function called XOR it is called exclusive OR. So, in case of OR if anyone is high then we are going to say that output is high. But in case of XOR we say that output is high provided either of them is high. So, in that particular case if both are high in that particular case output is 0. So, either of them a and b is high then output is high. So, similarly XNOR is nothing but the invert of XOR. So, exclusive OR and exclusive NOR so you just see the function value is just invert of this particular XOR gate.

Now, you just see that here I am just mentioning some of the gates like that we are having other gates also. So, in that particular case that NOT it is a unary gate we are having only one input and depending on that we are having the output, but other gates are your binary gates because we are having two inputs depending on this input we are going to have output. So now you just see that we are having two inputs, now here I have such some of their examples and how many such type of gates we can have if you are going to take two inputs. So, it is basically you can have several combination and if you look into all those combinations say these are the 4

different possibilities that we are having either both are 00, 01, 10 and 11 these are the only possible combination for 2 input gates. So, we are having 4 possible input combination and for these 4 possible in input combination what we can say that either all outputs are 0, all outputs are 1 or maybe some of them are 1 and some of them are 0. So, in that particular case you can say that we are going to get total 16 different combination; that means, 16 different function or binary function can be implemented if we are having 2 inputs.

Now secondly, these gates can be extended for more input driven set if I am going to look for a gate where I am having 3 input over here say 3 input AND gate. So, these gate can be visualized like that first I am going to have and combination of first 2 and secondly, the whatever output you are getting that will be combined with the third input. So, like that we are going to get the effect of 3 input AND gates or may be 3 input AND gates can be implemented with the help of electronic circuit. So, this is possible because these particular operations are associative; that means, $(ab).c = a(b.c)$. So, since it is associative, so such type of expansion is always possible.

(Refer Slide Time: 17:25)

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\bar{A}	$A \cdot B$	$\overline{A \cdot B}$	$A + B$	$\overline{A + B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table border="1"> <tr><td>A</td><td>0</td><td>1</td></tr> <tr><td>\bar{A}</td><td>1</td><td>0</td></tr> </table>	A	0	1	\bar{A}	1	0	<table border="1"> <tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>B</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>$A \cdot B$</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>	A	0	1	0	1	B	0	0	1	1	$A \cdot B$	0	0	1	1	<table border="1"> <tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>B</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>$\overline{A \cdot B}$</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> </table>	A	0	1	0	1	B	0	0	1	1	$\overline{A \cdot B}$	1	1	0	0	<table border="1"> <tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>B</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>$A + B$</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>	A	0	1	0	1	B	0	0	1	1	$A + B$	0	0	1	1	<table border="1"> <tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>B</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>$\overline{A + B}$</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> </table>	A	0	1	0	1	B	0	0	1	1	$\overline{A + B}$	1	1	0	0	<table border="1"> <tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>B</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>$A \oplus B$</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table>	A	0	1	0	1	B	0	0	1	1	$A \oplus B$	0	1	1	0	<table border="1"> <tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>B</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>$\overline{A \oplus B}$</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> </table>	A	0	1	0	1	B	0	0	1	1	$\overline{A \oplus B}$	1	0	0	1
A	0	1																																																																																																					
\bar{A}	1	0																																																																																																					
A	0	1	0	1																																																																																																			
B	0	0	1	1																																																																																																			
$A \cdot B$	0	0	1	1																																																																																																			
A	0	1	0	1																																																																																																			
B	0	0	1	1																																																																																																			
$\overline{A \cdot B}$	1	1	0	0																																																																																																			
A	0	1	0	1																																																																																																			
B	0	0	1	1																																																																																																			
$A + B$	0	0	1	1																																																																																																			
A	0	1	0	1																																																																																																			
B	0	0	1	1																																																																																																			
$\overline{A + B}$	1	1	0	0																																																																																																			
A	0	1	0	1																																																																																																			
B	0	0	1	1																																																																																																			
$A \oplus B$	0	1	1	0																																																																																																			
A	0	1	0	1																																																																																																			
B	0	0	1	1																																																																																																			
$\overline{A \oplus B}$	1	0	0	1																																																																																																			

Complete Set of Operators: NOT, AND, OR

Universal Gates: NAND/NOR

$\bar{B} \cdot A + B \cdot A$

Now, we are talking about this particular logic gate here we are having two terms called complete set of operators and I think you may be knowing about these things so the complete set of operators are nothing but AND, OR and NOT gate. If we have these 3 gates then, all others can be all other gates can be implemented with the help of these 3 gates. There is a simple example you just see that if I am going to talk about this particular XOR gate. What is

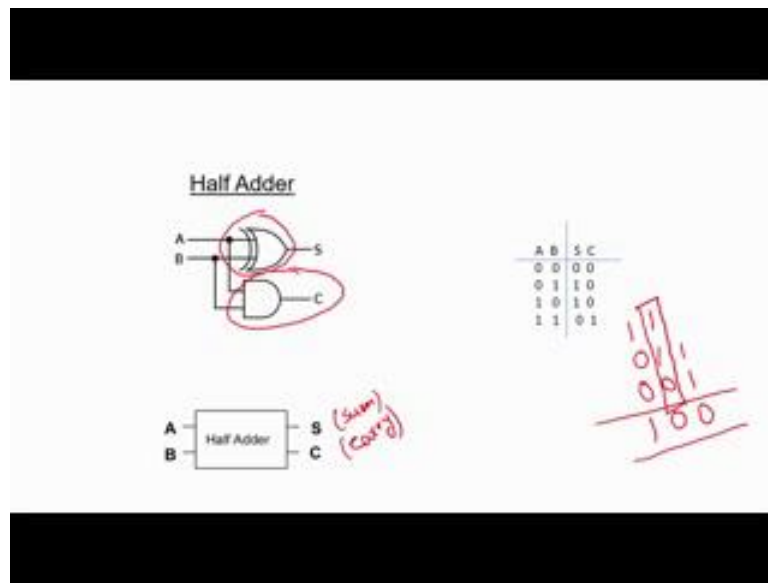
the behaviour? If 0 and 1 output is 1 and 1 and 0 output is 1 so that means what I can say that it is basically nothing but $\bar{B}A$. So, this is B is 0, A is 1 at that particular point the output value is 1 similarly either B or \bar{A} . So, this is nothing but this expression is going to give me this particular XOR combination. Now, just see if we don't have this particular gate then what will happen with the help of NOT, AND, and OR gate I can have the effect of this particular gates.

So, this is \bar{B} , \bar{A} we are using this particular NOT gate then $\bar{B} \cdot A$; that means, you are using this particular AND gate we are having two AND combination and they are now combined with this particular OR gate. So, you just see with NOT, AND, and OR all circuits can be implemented so that's why you are going to say that this is the complete set of operators.

Along with that we are having another terms called universal gates. So, NAND and NOR are treated as universal gates, Why you say these are the universal gates? Any digital logic circuit can be implemented with the help of only NAND gate or only NOR gate. So, this is the things that we are having that is why you said these are the universal gate. Now, as an assignment you just think how to be implement this particular function with the help of only AND gate, NOR gate or NAND gate just take as a assignment and just see how the circuit can be implemented with the help of only NAND gate or with the help of NOR gate.

Now, we are going to see some of the building blocks that will be used while constructing our computers. So, one thing is first circuit I am going to talk about here is your half adder this is basically adding of two numbers and these are the addition of binary numbers because we can represent only 0 and 1 that's all, high voltage means 1 and low voltage means 0. In that particular case that behaviour of half adder can be represented with the help of this particular truth table, so we are having two input A and B , and two output sum bit and carry bit S represent for sum and C represents for carry.

(Refer Slide Time: 20:14)



So, this is basically when both A and B are 0 then my sum is 0, when A and B is 1 then sum is 1, when A is 1 B is 0 then sum is 1 and when A is 1 and B is 1 I am going to get sum is 0, but it is going to generate a carry called 1. So, this behaviour for S we need one circuit, for C we need another circuit.

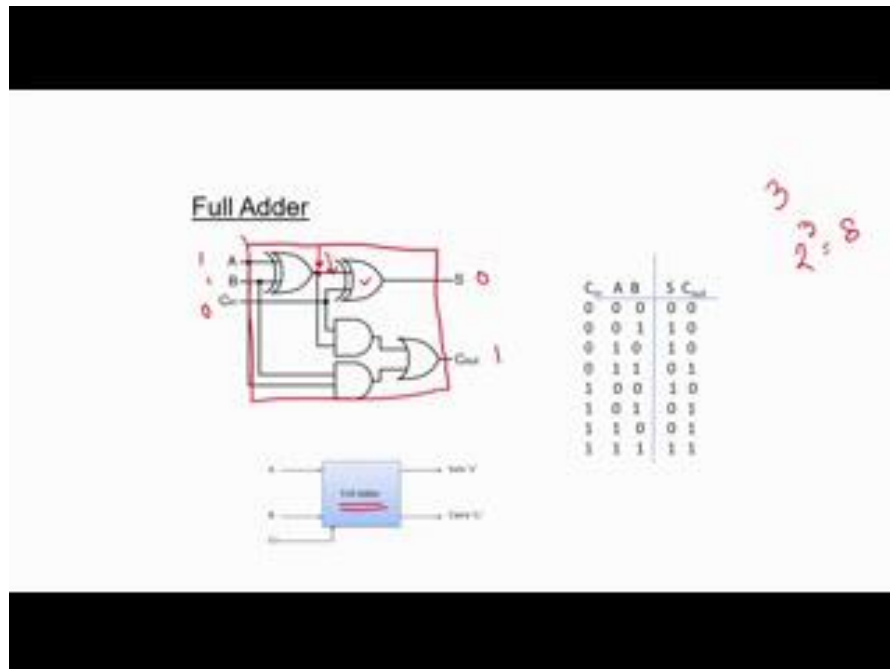
Just look into the behaviour of S , here we are going to say that this is nothing but the exclusive OR so that means this exclusive OR gate is going to give me the behaviour of sum function and if you see that C it is nothing but AND combination and both are 1 then output is 1. So, in that particular case this AND gate is going to give me this particular carry function.

Now, here I am talking about a half adder when we talk about half adder necessarily that something will come to your mind; that means, there may be some other adder also. Let's see this is an adder circuit which can add a single bit number now just consider about sum 3 bit numbers then what will happen, we are going to add it so 1 and 1 is 0 and it is going to generate carry 1, then 1 and 1 we are going to add together then output is 0 and it is going to generate 1 carry. Now 1, 0 and 0 is going to give you 1, so this is basically the result is your 100.

Now, the first bit is 1 you just see that if I am going to give A and B I can get the sum bit along with that I am going to get that carry bit. But when I am coming to the next bit now, what will happen you just see that I have to take decision the functional value output of this function depends on 3 inputs; that means, you have to take care of both the input A and B along with

that the carry that is coming out from my previous bit. So, for that we are having another circuit we call this is the full adder.

(Refer Slide Time: 22:28)



So in case of full adder, now you just see that we are having 3 input A , B and carry in C_{in} the C_{in} is coming from the previous bit position. So, it is coming from here and we are going to get this C_{in} and when we add them together it will generate a carry out C_{out} like this one. So, we are having this particular C_{out} . So, the behaviour of this circuit can be represented with the help of this particular truth table.

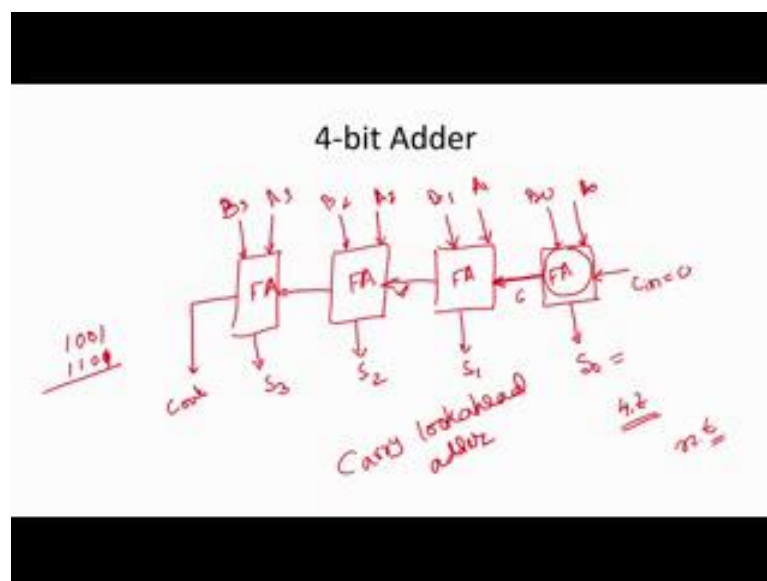
Now, since we are having 3 inputs over here. So, total input combination will be your 2^3 which is your 8. So, this is the maximum number of combination that we have. So, we have cited all those possible combination over here and depending on the behaviour of addition we are going to find out what will be the my output function. Now, once we have this particular output function then we can implement with the help of this particular circuit. Now, you just see that, now this complete circuit will be treated as my full adder and finally, once you give the signal over here say 110 finally, we are going to get the output as your 1 and 1 sum is 0 carry is 1.

Now, we have to see one more things. Now, said these are some logic gates or electronic component when you put signal 1 and 1 immediately we are not going to get the result over here because this electronic components are having some delays, it need some times and that

time whatever time it is required we say this is the propagation delay of that particular block, here we are simply using an exclusive OR gate. So, this XOR gate will take some time to give me the final stable output over here. So, once we get a stable output then only we can work with this particular second gate because 0 is coming, C_{in} is coming immediately as soon as I am giving the input, but the second input to this particular XOR will come after some unit of time only which depends on the propagation delay of this particular gate and this second gate will also have some propagation delay so finally, we are going to get a correct result. So, this time we have to consider and we are going to say that the final output we are going to get after some amount of time once you give the stable input and this time depends on the propagation delay of the components of that gate so that means to get the result of a full adder is going to take some time we have to consider that particular time also.

So, this is the internal circuit. So, as a block represents we can say that this is full adder I am having 2 input A, B along with that we are having that carry in and it is going to give me 2 output sum bit and carry bit. So, this is the block diagram, but what is there inside this particular full adder if we look in there we are going to have this particular circuit only. Now, we know or we have seen the behaviour of half adder and full adder. Now, if we are going to construct a 4 bit adder then how we are going to construct it; that means, we need to add 4 bits so that means we need 4 adder or here we can use the full adder even.

(Refer Slide Time: 25:51)



So, what I can say that this is my 4 full adder, so here I am having two input $A_0, B_0, A_1, B_1, A_2, B_2, A_3, B_3$. Now, what will happen this first full adder is going to give me the sum bit and along with that it is going to be give me 1 carry out also that carry out will go as a carry into the next full adder. So, this is your S_1 and it will go that carry out will go as an carry in this is your S_2 , this is your S_3 and along with that I am going to get 1 carry out also. Since least significant bit not having any carry input so that carry input bit we will set to 0, on the other hand this full adder can be replaced by half adder also. Now, this is the 4 bit adder. Like that if I need an 8 bit adders I can cascade 8 full adder I am going to get the circuit.

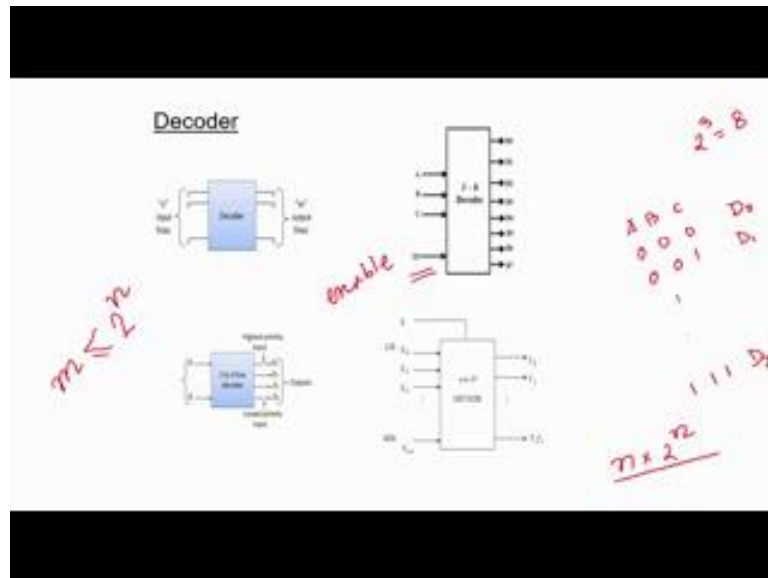
Now, you just see I am talking about the propagation delay. Now, this full adder is having some propagation delays. Once I am giving this A_0, B_0 and C_{in} it will take some time and we are going to get the output over here and along with that this C bit we are going to get. Now, I am giving this 4 bit number say 1001 and 1100 in that particular case or I consider that second full adder cannot perform a job immediately as soon as I am giving A_1 and B_1 it is a it needs to wait for the stable output in this particular carry out of the first full adder.

Once we are going to get the stable output then only this particular second full adder can perform its output, so like that second full adder is going to give me some output, the third full adder need to wait for that particular stable output. So, in that particular case you just see that if the propagation delay of the first adder is some time t then total time that required to get a correct output of this full adder will be maybe your 4 times of t because second full adder is going to give me the correct result after first after t unit of time. Second full adder is going to give me a stable output after $2t$ time. So, like that it is taking the $4t$ time. Now like that if you are going for n bit data; that means, the total propagation time will be your nt , where t is the propagation delay of 1 full adder.

So, you just see that nowadays we are talking about say 32 bit computers, 64 bit computers that means it will take lot of time to give me the output. So, for that the simple circuit we are not going to use for that we will use another circuit modification of this particular adder circuit which is known as your carry look ahead adder. So, here we are not going to discuss about this things as an information I am giving you that we are having another circuit where it is known as your carry look ahead adder and carry's will be generated at the same time and simultaneously it will be propagated ok carry generation and carry propagation so, carry look ahead adders. So, with the help of this carry look ahead adder we can reduce this particular propagation delay of the adders. So, this is some bit of information I am giving you so that are

if you are interested then you can just look for some any book on digital logic where adder has been discussed thoroughly.

(Refer Slide Time: 30:10)

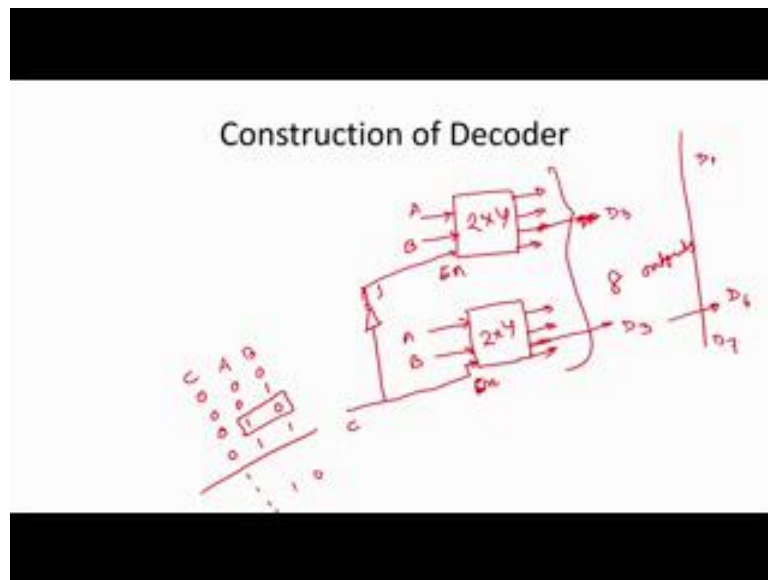


Now, the second circuit is your decoder you are having another building block called decoder. Here we are having n input lines and m input output lines and the m whatever m output lines we are having this is basically it is less than equal to 2^n . So, if we are having n input lines maximum output lines we are going to get as your 2^n . And what is this circuit behaviour? Depending on the input combination only one of the output line is high. Ok so, here you just see that we are having 3 by 8 decoders we are having 3 input lines and we are having 8 output lines D_0 to D_7 . So, this 3 input lines A, B, C . Now, what will happen since we are having 3 input lines so, what will be the total number of input combination this is your 2^3 this is equal to 8. Depending on one combination generally one of the output line is high. So, basically what will happen when both are all are 0 then D_0 is high when that one signal is high that C is high in that particular case I can say that D_1 is high like that when all combination is 1 then you said it this is your D_7 is high.

So, depending on the input combination only one of the output signal is high. So, this decoder we are going to use for selecting some of our requirements or some of our function. Say if we are having some choices that that particular instance of time we have to select only 1 in that particular case we are going to use this particular decoder. So, this is a 3×8 decoders, like that we are going to get $n \times 2^n$ decoder where n is the input line and 2^n is the output lines. In this

particular decoder in most of the cases we are having an signal called enable signal EN is nothing but enable, that means when this enable signal is high then only it is going to behave like a decoder otherwise it is not going to work may be all output will be 0. So, this is with enable line we are going to have a decoder.

(Refer Slide Time: 32:39)



Now, this enable line help us to construct some of the decoder say if you are having a decoder say I am having with me say 2×4 decoders and I say that I am having it is having 2 input lines A and B, but I need a say 3×8 decoders then what I can do, I can use two 2×4 decoders. So, these are the output lines 4 output lines we are having and these are the 4 output lines that we are having for the second decoder. So, total 8 output lines we will get.

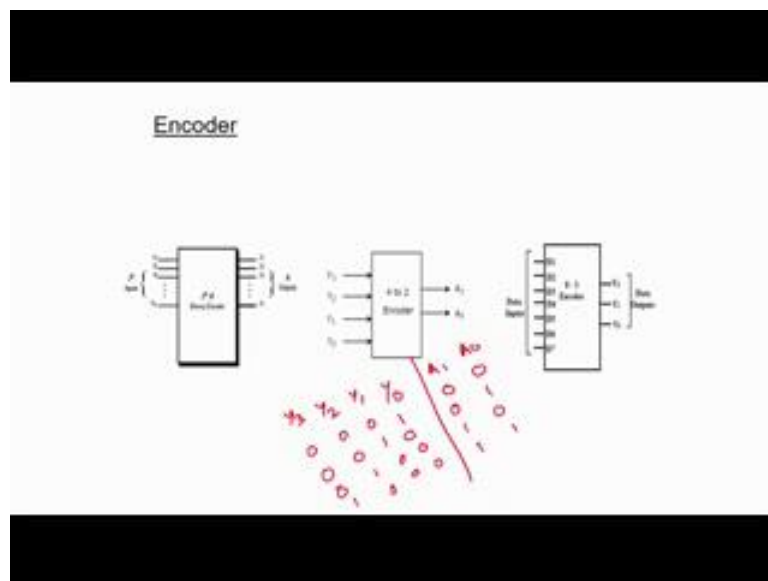
Now, here in both the cases I am going to give A 0, A and B. Now, depending on this particular A and B say now A and B is having 2 combination only 4 combination 00, 01, 10 and 11. When input say I am giving say 1 and 0 then what will happen this particular line of say D_3 of this particular decoder and D_3 of this particular decoder will be selected because this is the combination. Now, we have to select one of these two outputs. So, for that what will happen? We need another input, say I am having this particular C input then what we are going to do we will use an inverter and two this things over here this is the enable line and this is the enable line over here. Now, we just see that when I am having C when $C = 0$ then what will happen we will see that we are going to this is $C = 0$. So, enable line is 0 over here so you are not selecting this one since $C = 0$. So, here I am going to get 1. So, in the particular case we are

selecting this particular decoder; that means, you are selecting this particular decoder output line D_7 and all these things will be 0.

Now, similarly when $C = 1$ again I am going to get 4 combination. So say 10 then what will happen in that particular case since $C = 1$ here I am going to get 0 so we are not selecting this particular 4 lines and we are selecting only that particular D_3 and it is nothing but now we can say that this D_3 is going to act as my D_6 because we are going to have from D_0 to D_7 . So, like that we can use a decoder or say you can use multiple decoder to construct a bigger decoder and it is possible due to this particular enable lines.

Now, we are having another circuit call encoder. So, encoder is the reverse of decoder in case of decoder we are going to decode from n line to 2^n line, in case of encoder we are coming from 2^n lines to n lines. So, total we are having 2^n input lines and we are having n output lines. So, this is a 4×2 encoder.

(Refer Slide Time: 35:38)



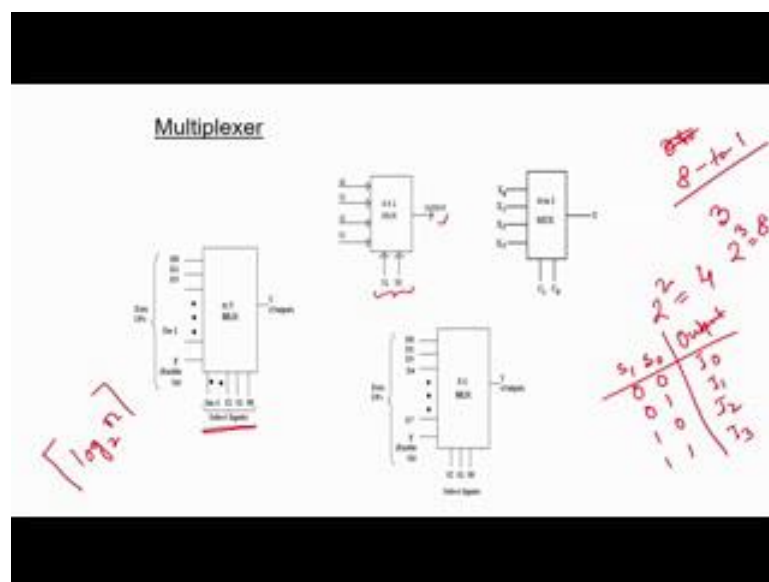
So, depending on this scenario, now you are going say either A_0 or A_1 will be high. Now, what we can do, say when I am going to say that this Y_0, Y_1, Y_2, Y_3 . Now, at any point of time say any one of the input line is high. So, this is basically say I am having this particular pattern or maybe this is.

So, these are the 4 inputs then what basically I can say that this is your. So, this input says that I am having 4 inputs. Now, depending on that I am going to encode it and in that particular

case I can say that this one Y_0 is 0 we will say that it is going to represent 0. So, both A_1 and A_0 will be 0, when Y_1 line is high then in that particular case we will say that it is going to represent 1 so this will be your 01, when Y_2 line is 1 then it may be represented by 2 and when Y_3 line is high it is represented by binary 3. So, this is the way we can encode it. So, encoder is reverse of your decoding circuitry. So, here we are having 2^n input lines and we can get n output lines.

So, here what happens we are just discussing those things in block level how to implement it these are very simple combinational logic circuit. You can take any book on digital logic and you can see that how it is done and this is very simple circuit I am telling you with the help of AND gate it can be done. So, this is encoder. Like that another building block we have which is known as your multiplexers. So, it is basically going to multiplex or masking some of my input signals. So, it is basically like that we are having several signals at any point of time we are going to take only 1 signal out of these particular n possible signals. So, here it is something like that if we are having n input signal we are going to put it to 1 output signal. So, it is called $n \times 1$ mux.

(Refer Slide Time: 37:52)



Now, how you are going to select it for that we need some select line or control lines, that control lines depends on this particular number of input lines. If we are having n input lines then number of select line will be your $\lceil \log_2 n \rceil$ ceiling of that. Now, we will see these things will a small example. Say I am talking about a 4×1 mux. Now, we are having 4 input lines I_0 ,

I_1, I_2 and I_3 depending on my requirement we are going to take only one of this particular input signal and going to put it into the output signal which is known as your masking the other signal and I am going to put the in 1 particular input lines to the output line. To select this particular 4 input what happened we need some select line or control line. So, here we are having two select line S_0 and S_1 .

So, we know that $2^2 = 4$; that means, depending on the combination of this particular two select line we are going to select one of this particular input lines. So, basically I can say that this is your S_0 and S_1 . So, what is the combination of those particular select line? That maybe 00, 01, 10 and 11. Now, what will happen depending on this particular select line what will be the my output we are going to select one of this particular input signal.

So, say that when S_0 and S_1 is 0 then I can say that we are going to select that input line I_0 and it will be transfer to the output side, when it is your 01 then I_1 , then 10 I_2 and 11 is your I_3 . So, this is the behaviour of multiplexer. From n possible input signal we are going to select only 1 out of those n signals, so that's why what is the input, output and select line relationship? We are having n input lines, we are having 1 output lines. To transfer any one of this particular input lines to the output lines we need select signal or control signal. What is the number of control signal it is your $\lceil \log_2 n \rceil$. If we are having 8 to 1 mux then what will happen we need 3 select line because $2^3 = 8$. So, this 3 select line will have combination from all 0s to all 1s 8 possible combination. So, we are going to select any one of these particular input lines and I am going to put in to the output. So, this is a multiplexer circuit again it is a combinational circuit and very simple circuit.