

Digital Circuits
Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 25
Decoders, Multiplexers, PLA (Contd.)

We can look into some chips that realize these multiplexers.

(Refer Slide Time: 00:21)

MSI Quad Two-input Multiplexer

- The 74LS157 contain of quad two-input multiplexers, $I_{0a}/I_{0b}/I_{0c}/I_{0d}$ and $I_{1a}/I_{1b}/I_{1c}/I_{1d}$.
- The logic symbol and truth table is shown in Figure.
- Notice that each of the four multiplexer shares a common data select line and a common *Enable*.
- Each multiplexer has only one data select input because there are only two groups of inputs to be selected.

The slide includes a diagram of a 2-to-1 multiplexer with inputs I_0 and I_1 , a select input S , and an output Y . The truth table shows that when $S=0$, $Y=I_0$, and when $S=1$, $Y=I_1$.

At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL Online Certification Courses, along with a small video feed of the professor.

For example, this 74157 chip: so it has got Quad Two-input multiplexers; that means, it has got in a in the chip there are 4 multiplexers. So, if this is the full chip, so there are four multiplexers ok. So, they are all Two-input multiplexers that there are 4 such multiplexers. And this multiplexers, so they have input a 0 a 0 b so, I_{0a} I_{0b} I_{0c} and I_{0d} has the 0 inputs and I_{1a} I_{1b} . So, in the first multiplexers; so this is multiplexer numbers 0. So, it has got I_{0a} and I_{1a} as input.

The second multiplexer is multiplexer 1. So, it has got I_{1a} and sorry I_{0b} and it has got I_{0b} and I_{1b} , so that way then other. So there are 4 multiplexer, one is a b c and d. So, any multiplexer, it has got the so if the multiplexer a, it has got input I_{0a} and I_{1a} multiplexer b, it has got inputs I_{0b} and I_{1b} . So, like that and the select lines are same; so for all of them, so there is one select line only. So, we can just the select line and that enable lines. So, they are common so, for the all multiplexers.

(Refer Slide Time: 02:00)

Inputs		Outputs			
\bar{E}	S	Y_a	Y_b	Y_c	Y_d
H	X	L	L	L	L
L	L	I_{0a}	I_{0b}	I_{0c}	I_{0d}
L	H	I_{1a}	I_{1b}	I_{1c}	I_{1d}

Logic symbol and truth table for 74LS157 multiplexer.

So, will see how this is working. So, for that we need to look into the corresponding truth table. So, this is the symbol that we have. So, there is a select line, there is an enable line. So, for this multiplexer quad multiplexer to operate, this E line must be equal to 0, then only this multiplexer will operate. Now, this is the actually the E bar. So, if E bar equal to high, then irrespective of the value of S the select line. So all this outputs are equal to low.



However if so, if e bar is low that is then activates enable signal is given to 7 4 1 5 7, then if the select line is low, then in the Y a output I will get I 0 a, in the Y b output I will get I 0b, Y c it get I 0c and y d I will get I 0d. So, essentially you can say that the structure is something like this.

So, I have got I have got 4 multiplexers. So, I have got 4 multiplexers connected like this ok. So, I have got this 4 multiplexers and then the first multiplexer has got I 0a and I 1a I 0a and I 1a, the second multiplexer has got I 0b and I 1b, third multiplexer has got I 0c and I 1 c and the fourth multiplexer has got I 0d and I 1d and all for all this multiplexer the select line is same. So, that is coming from S. So, that is a select line. So, that is same. And this output is called Y a, this output is called Y b, this output is called Y c and this is called Y d and there is an enable line of course. So, this is the structure, the logically this is the structure that we have inside the multi inside the chip.

(Refer Slide Time: 04:11)

MSI Quad Two-input Multiplexer

- \overline{E} Input is *LOW* – allows the selected input data to pass through to the output.
- \overline{E} Input is *HIGH* – will disable the multiplexers, all of the outputs will be *LOW*.
- When $\overline{E} = 0$ and $S = 1$, the Y outputs will follow the set of I_1 inputs, that is $Y_a = I_{1a}$, $Y_b = I_{1b}$, $Y_c = I_{1c}$, and $Y_d = I_{1d}$.



Now so, E bar input is low. It allow the selected input data to pass through the pass through to the output and E bar input is high, it will disable the all the multiplexers and all outputs will be low. When E bar equal to 0 and S equal to 1, the Y outputs will follow the set of I 1 inputs where y a equal to I 1a y b equal to I 1b etcetera. Similarly when S equal to 1, it will be following the 0 inputs; so in that case Y a will be I 0 a, Y b will be I 0b like that. So, that is the standard multiplex.



(Refer Slide Time: 04:45)

LOGIC FUNCTION GENERATION USING MUX

Exercise 1:
Implement the logic circuit function specified in the table given below by using 74LS151 8-input data selector/multiplexer.

Input			Output	
A2	A1	A0	Y	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	1	3
1	0	0	0	4
1	0	1	1	5
1	1	0	1	6
1	1	1	0	7

Q:1

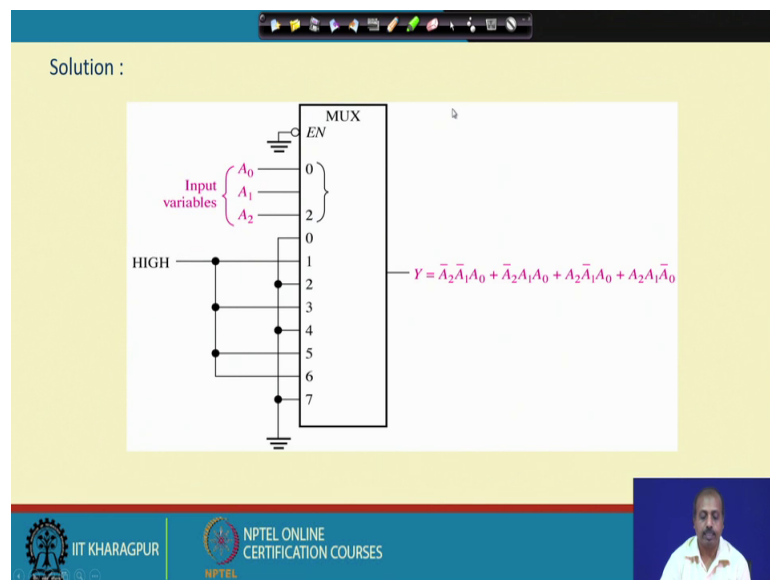


Now we will be looking into some technique by which we can realize some logic circuit function using some multiplexers. So, will take has an example 7 4 1 5 1 which is an 8 input multiplexer. So, it is 8, so 8 8 is to 1 multiplexer we can say. So, we normally we also for the multiplexers, we write it like 8 is to 1 multiplexer that is there are 8 input lines, 3 select lines and 1 output line.

Now, what we do is that so this is suppose this is the function that we want to realize. So, this is the combinational function, 3 input combinational functions. So, for the combination 0 0 1 0 1 1 1 0 1 and 1 1 0, the output should be 1 and for the rest of the places the output will be 0.

So, what we do first for doing it for realizing the circuit is that we note down the corresponding decimal value of the different min term ok. So, the first min term is 0, second is 1, third is 2 like that. Now we note down the places where the value is equal to 1 and accordingly we get the circuit. Like here you see that this function can be realized like this. So, what this enable line is made low, so that this multiplexer is enabled.

(Refer Slide Time: 06:07)

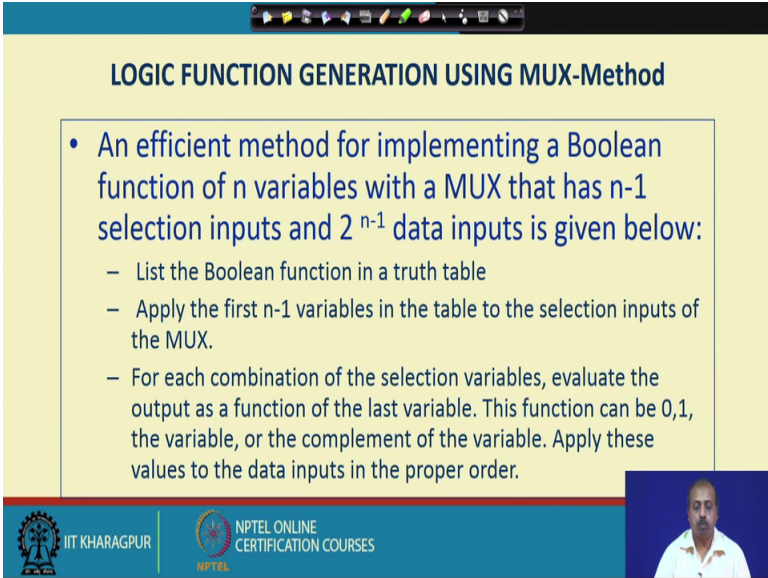


Now, this $A_0 A_1 A_2$; so they are connected to the select lines of the multiplexer ok. So, 0 1 and 2 and then if you look into to this place, you see for 1 3 5 and 6. The output is 1 and rest are 0 for 1 3 5 and 6 it is equal to 1. So, for data input 1 3 5 and 6, so they are tied high and this data input 0 0 then 2 and 4 and 7, so 0 2 4 and 7 for them it is low. So, it is for 0 2 4 7, they are tied low.

Now, the way it operates is if you apply some value here $A_0 A_1 A_2$ some pattern here, depending upon the value one of the inputs lines will one of this data inputs lines will be selected to the output and if it happens that it selects a line from this 1 3 5 and 6. Then you will get a high here or 1 here, but if it selects if the values are such that it is selecting one of this data input 0 to 4 and 7 then you will get a 0 there.

So, this way using this multiplexer, we can realize any combinational logic ok. So, we can just, what we need to do is sum of the data inputs are to be tied high, some of the data input are to be tied low. And this combinational function variable, so they should be tied to the select lines of the function or select lines of the multiplexer and definitely the enable line has to be activated ok.

(Refer Slide Time: 07:56)



LOGIC FUNCTION GENERATION USING MUX-Method

- An efficient method for implementing a Boolean function of n variables with a MUX that has $n-1$ selection inputs and 2^{n-1} data inputs is given below:
 - List the Boolean function in a truth table
 - Apply the first $n-1$ variables in the table to the selection inputs of the MUX.
 - For each combination of the selection variables, evaluate the output as a function of the last variable. This function can be 0,1, the variable, or the complement of the variable. Apply these values to the data inputs in the proper order.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, how can I do these things? So, this whole operations, which can be summarize like this. So, these an efficient method for implementing Boolean function of n variables with a multiplexer that has n minus 1 select input and 1 that is and 2 power n minus 1 data input. So, it is like this that in this previous example in this example, what has happened is that I have got 3 input it is a 3 input function ok. So, this is a 3 input function and for that we needed one 8 is to 1 multiplexer.

So, but if I do that then, what happens is that many times the multiplexer size becomes too high. For example, in my library or I may not have this 8 is to 1 multiplexers available, may be only 4 is to 1 multiplexer are available. That means, what I what is

needed is I need to do this thing. The I have got a I have got an n variable function, but I will I will be realizing using then using multiplexers that has got n minus 1 select lines that is 2 power n minus 1 data lines.

So, what we do for this is that we the first make the truth table and the first n minus 1 variables of the table, so they are connected to the select inputs of the multiplexer. And for each combination of select variables evaluate the output has a function of the last variable. And accordingly we put the variable has 0 1 or the variable or the variable itself. There is an example.

(Refer Slide Time: 09:34)

LOGIC FUNCTION GENERATION USING MUX-Example 2

Implement the Boolean function $F = x'y'z + x'yz' + xyz' + xyz$ using a suitable MUX

x	y	z	F
0	0	0	0
0	0	1	$F = z$ 0
0	1	0	$F = z'$ 1
0	1	1	0
1	0	0	$F = 0$ 2
1	0	1	0
1	1	0	$F = 1$ 3
1	1	1	1

So, suppose this is the function that is given to me. So, $F = x \text{ bar } y \text{ bar } z + x \text{ bar } y z + x y z \text{ bar} + xyz$; so what we do? We have got so this is a 3 variable function. So, you can straight way realize it using 8 is to 1 multiplexers. So, like this. So, I can do it in this fashion, using an 8 is to 1 multiplexer. So, this is the select lines. So, I apply x y and z and this data lines, so these are the data lines 0 1 2 3 4 5 6 and 7 and then I know that for 1 2 6 and 7, the output should be 1. So, for 1 2 6 and 7, I tie them together to logic high. So, this is tied to logic high and 0 3 4 and 5, they should be tied to low.

So, what I do? I take these lines and tie them to low. So, this is made 0. So, this way using, so this is my function this is my F. So, by using this 8 is to 1 multiplexer, so I can definitely realize it. But how can I do it using 4 in put multiplexer.

So, for doing it with 4 input multiplexer, we have to proceed like this. So, we make the truth table and then we first of this x and y, so these inputs we apply directly to the select lines S₀ and S₁ and for the third input, I figure out like this. First, so for if I look into this first two terms, so 0 0. So, whenever this x y equal to 0 0 that forms 1 group, then 0 1 it forms the second group, 1 0 x y equal to 1 0 from the fourth third group and 1 one from the forth group. So, these are the 4 data inputs like this will correspond to the n put pattern S₀ x x y equal to 0 0, this will correspond to 0 1, this will correspond to 1 0 and this will correspond to 1 1.

Now, if I look into the portion of the truth table where it is 0 0, so that portion of the truth table and if I do a minimization in terms of the third variable, so I will get F equal to z. So, that way in this line I apply z. Next for this is 0 1 for x y equal to 0 1, so I get here as z bar ok. So, if you look into this function here, so that is z bar. So, that is I getting as z bar. Then this part is all 0. So, it is 0 and it is all 1. So, this is again one.

So, I take a 4 is to 1 multiplexer then of course, I need some additional circuitry. So, what is what is actually needed, we were needing is apart from the multiplexer, I need an inverter here for getting this z bar ok. For getting this z bar, I am needing another inverter, but otherwise I can do it like this.

(Refer Slide Time: 13:19)

LOGIC FUNCTION GENERATION USING MUX-Example 2

Implement the Boolean function $F = x'y'z + x'yz' + xyz' + xyz$ using a suitable MUX

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

The slide also shows a handwritten logic diagram of the function $F = x'y'z + x'yz' + xyz' + xyz$ and a 4x1 MUX circuit with select lines y (S₀) and x (S₁), and data inputs 0, 1, 2, 3. The output is F.

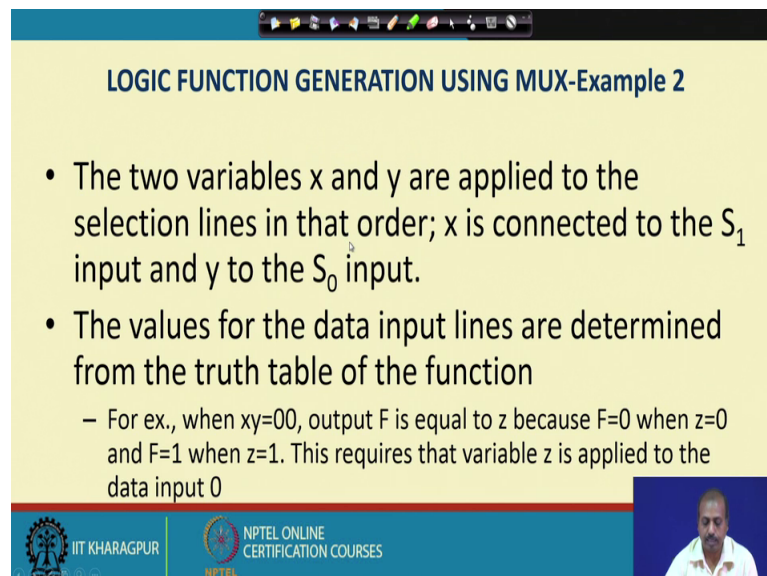
So, this way you can realize circuits using different number of multiplexers. For example, if I want to do the same thing using 2 is to 1 multiplexer, then what to do? So,

it is using 4 is to 1 multiplexer. So, if I have to do it using 2 is to 1 multiplexer, then this 2 is to 1 multiplexer, it will have only 1 select line and 2 data lines. So, on this select line, I apply x and this data lines are 0 and 1. So, I partition this truth table into two partitions. So, in one partition x equal to 0 other partition x equal to 1.

Now if I do a minimization, you see that here the function that I am getting is in this part, if you look into this part, the function that I am getting; so it is 1 when only 1 of y and z are equal to 1. So, this is nothing, but y XOR z and in the lower part, in the lower part I am getting the function for this part so, whenever this y is equal to 1, the output is equal to 1. So, for this part I have got the function y . So, what I can do? I can apply y here and I can take 1 XOR gate, I can take an XOR gate and take this y and z inputs to the XOR gate ok. So, I can do it like this. So, this is my F . So, if this is using 2 is to 1 multiplexer.

So in this way, you can have these multiplexers of lesser sizes to realize these combinational functions. So, if we are having same, if we are there is a n variable function. And if you have got a 2^n is to 1 multiplexer, then it is straight forward as it is reducing, then you can you have to do the grouping of this terms and accordingly you have to get the truth table. You have to get the multiplexer realized.

(Refer Slide Time: 15:15)



LOGIC FUNCTION GENERATION USING MUX-Example 2

- The two variables x and y are applied to the selection lines in that order; x is connected to the S_1 input and y to the S_0 input.
- The values for the data input lines are determined from the truth table of the function
 - For ex., when $xy=00$, output F is equal to z because $F=0$ when $z=0$ and $F=1$ when $z=1$. This requires that variable z is applied to the data input 0

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, next we will see so, two variables x and y have select line. So, this is the thing that we have discussing; when x y equal to 0, F we can see equal to z . So, we have also seen the 2 into 2 multiplexers like that.

(Refer Slide Time: 15:30)

LOGIC FUNCTION GENERATION USING MUX-Example 3

Implement the Boolean function $F=A'B'C'D+A'B'CD+A'BC'D'+AB'CD+ABC'D'+ABC'D+ABCD'+ABCD$ using a suitable MUX

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

So, this is another example where I have got this 4 variable function $A' B' C' D + A' B' C D + A' B C' D' + A B' C D + A B C' D' + A B C D' + A B C D$ and this I want to realize a plus all this. So, this is the 4 variable function, so I will need a 16 is to 1 multiplexer, If I want to do a directly. So, if I want to realizing it using 8 is to 1 multiplexer, then we can do it like this. So, we are applying this A B C to the select lines of the 8 is to 1 multiplexer and then we are dividing the truth table into regions of similar values of A B C.

So, first part ABC values are 0 0 0 and second part is 0 0 1 like that. And if you look into the corresponding function F, then for the first part F equal to D, second part also F equal to D, third part F equal to D dash, then F equal to 0. So, it goes like this. So, I can take help of a few signals a few lines and inverter to feed the data lines of the multiplexer and accordingly I can get the 4 input function realize using 8 is to 1 multiplexer or 8 by 1 multiplexer.

So, this way we can use multiplexers to realize logic functions. So, another technique by which we can realize this multiplexer base circuit is where we do not go for higher degree multiplexer, but many times we want to realize the circuits using 2 is to 1 multiplexer only. So, if you want to do that say I have got some function F which is a 3 variable function, So, x y and z and this is the function is given by say x y plus x bar z like this ok.

(Refer Slide Time: 17:14)

Shannon Decomposition

$$f(x_1, x_2, \dots, x_n) = \bar{x}_i \cdot f(x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n) + x_i \cdot f(x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

$$f(x, y, z) = \bar{x} \cdot f(x=0, y, z) + x \cdot f(x=1, y, z)$$

$$f(x, y, z) = \bar{x} \cdot z + x \cdot y$$

Diagram: A 2-to-1 multiplexer with inputs 0 and 1, and a select line x. The output is f(x, y, z). The function is decomposed as f(x, y, z) = \bar{x} \cdot f(x=0, y, z) + x \cdot f(x=1, y, z) = \bar{x} \cdot z + x \cdot y.

Now, if you want to realize it using 2 to 1 multiplexer, then we can do it like this. So, there is one decomposition which is known as Shannon Decomposition. So, this Shannon Decomposition in the most generic form it tells that if I have got an n variable function x_1, x_2 up to x_n then this can be decomposed around the variable x_i , and the decomposition is given by \bar{x}_i into f . Where, these other variables remain unchanged only x_i is set to be equal to 0, x_i is set to be equal to 1 and then plus x_i into f of $x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n$; that means, in the function f you put x_i equal to 0 in 1 case and x_i equal to 1 in the other case.

So, if in both cases the function f reduces to a function of $n-1$ variables because 1 variable has been fixed to some constant. So, that variable is done. So, you have got a function of $n-1$ variables. And then we put \bar{x}_i into $f(x_1, x_2, \dots, x_n)$ with x_i equal to 0 and then plus x_i into this. So, this is known as Shannon Decomposition.

So, if you apply the Shannon Decomposition, so essentially you see what happens is that if I have got this function F ; then if I have got a 2 to 1 multiplexer, then I can do it like this. So, this is the input 0, this is the input 1 and here in the select line I apply the variable x_i and when x_i is equal to 0. So here I have to somehow realize the function. The first function $x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ and here I have to realize the function $x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n$. Then again these functions can be decomposed using Shannon Decomposition around another variable. So, so that way it goes on.

So, let us take an example say this function that we have talking about. So, if we decide that we will decompose it around single variables say x , then what happens is that $f(x, y, z)$ it can be written as $\bar{x}f(0, y, z) + xf(1, y, z)$. This is on the definition of that Shannon Decomposition.

Now in this function, if I put x equal to 0; so what you get is $f(0, y, z)$. So, it says that it is \bar{x} into $f(0, y, z)$ where $f(0, y, z)$ is this function and it tells that if you for the second case if you put x equal to 1 here; so you are getting $f(1, y, z)$. So, this is x into $f(1, y, z)$. So, it tells that you take a multiplexer 2 is to 1 multiplexer and then if you apply x as the select line, then you will be getting the so you apply, so this is 0 and this is 1. So, you apply $f(0, y, z)$ here and $f(1, y, z)$ here. So, this is the realization of the function using 2 is to 1 multiplexer.

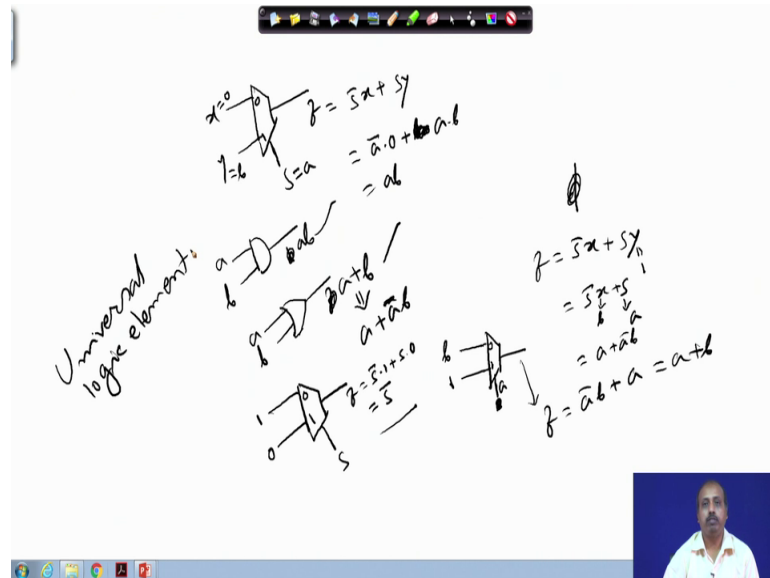
Now, if somebody instead of taking x has the variable around which you decomposed, so takes a y has the variable around which to decomposed. Then what happens is so, you gets a function realization has $f(x, y, z) = \bar{y}f(x, 0, z) + yf(x, 1, z)$. So, that is \bar{y} into if I put y equal to 0 here. So, I will get the function $f(x, 0, z)$ and then if I if I put y equal to 1, then this gives me $x + \bar{x}z$. So, that is nothing, but $x + z$. So, here I am getting $x + \bar{x}z$. So, by Boolean algebra so I can say it is y into $x + z$. So, it says that at the top level you have got this realization, if you apply y as the control input select input. Then you have got so, here I need to realize function $\bar{y}f(x, 0, z)$ and here I need to realize the function $x + z$.

So, do the same thing now. On this $\bar{y}f(x, 0, z)$, we apply Shannon Decomposition. So, if I apply Shannon Decomposition, then if I select x has the select line; if I put x has the select line like this, if I put x has the select line then when x equal to 0, so this is $f(0, 0, z)$ and when x equal to 1, so this is $f(1, 0, z)$. So, I can do it like this and for realizing this $x + z$ again, I can do it using Shannon Decomposition around x . So, when x equal to 0, so this is z and when x equal to 1 this is 1. So, this multi 2 is to 1 multiplexer based realization. So, it realize has the same function as this one, but you see depending upon the variable around which we decomposed in Shannon Decomposition in case you need single multiplexer, in another case you are requiring 3 multiplexer. So, that can happen.

So, that way any n input function it can be realized using only 2 input functions and this whenever I am requiring say some complimented input also so, sometimes so, you can also you can also do it using this input said inverter also like whenever we are having a 2

is to 1 multiplexer; so, 2 is to 1 multiplexer the function that we have is say x y and this is the select line s this is the output f . So, f is given by s bar x plus s y s bar s plus s y .

(Refer Slide Time: 24:25)



Now, you can use this structure to realize AND gate, how? So, for AND gate, I want that if I have got two inputs a and b , it should produce a 1 here. So, you see that if you it should so, it should give me the function a b . So, you see that if I apply a at S , if I apply a at s and b at y and if I make this x equal to 0, then this function transform to something like this. So, s bar that is a bar into 0 plus a into plus a into b . So, a bar into 0 is 0. So, you are getting a b .

So, this way I can realize 1 AND gate using this multiplexer 2 is to 1 multiplexer. Can I get an OR gate? So, for getting OR gate, OR function what I need is a b and this f should be a or b . This, a or b is can also be written as a or a bar b . So, you have seen that these to expressions are equivalent a plus b and a plus a bar b . So, what we do in this case? So, if I take this s as a , so if I take a say y equal to 1; so, I have I have got this expression f equal to s bar x plus s y . Now if try to draw if I put this y to be equal to 1, then this expression turns out to be s bar x plus s .

Now, if you set this s to be equal to a and this x equal to b , then this is nothing, but a plus a bar b . So, what I essentially get is that if I have a two input multiplexer, then on the line x I put b . So, this is 0 and 1 sorry 0 and 1 and this is my s . So, on the 0 line you put b and this y and the y input. So, if this is 1 this is equal to s , this is equal to a s equal to a .

Now, the function that you get here is f equal to $\bar{a}b + a\bar{b}$ that is nothing, but $a \oplus b$. So, I can get this OR gate also realized. So, can I get an inverter? So, if I have got this function so now this s ; so, this is 0 and this is 1. Now you see that if s is 0, the output should be 1. So, I make it 1 here and I can take the multiplexer and then the 0 input I connect 1 and 1 input I connect 0. So, accordingly I will get here, output function f has \bar{s} into 1 plus s into 0, so that is equal to \bar{s} . So, I am getting the inverter also. So, I am getting AND gate, OR gate and the inverter.

So, this 2 to 1 multiplexer. So this is also an universal logic element. So, it can realize all the logic circuit that we have we can think about ok. So, this is another universal logic element. Apart from the NAND gate, NOR gate that we have seen this 2 to 1 multiplexer is also an universal logic element.