

Digital Electronic Circuits
Prof. Goutam Saha
Department of E and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture-16
Multiplexer: Part I

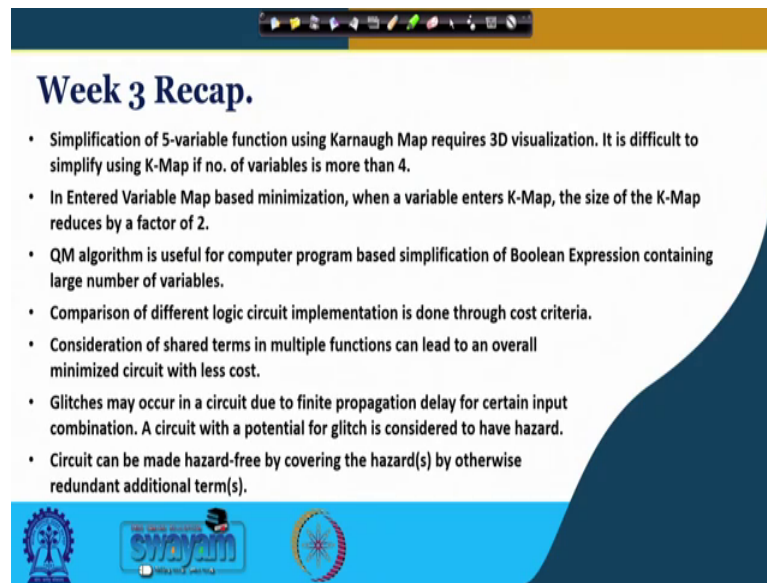
Hello everybody. We are in week 4 of this particular course in this class and also in the next class we shall discuss Multiplexer.

(Refer Slide Time: 00:22)



And in this class, we shall cover multiplexer basics and higher how to get a higher order multiplexer from lower order and its relation to Shannon's expansion theorem.

(Refer Slide Time: 00:32)



Week 3 Recap.

- Simplification of 5-variable function using Karnaugh Map requires 3D visualization. It is difficult to simplify using K-Map if no. of variables is more than 4.
- In Entered Variable Map based minimization, when a variable enters K-Map, the size of the K-Map reduces by a factor of 2.
- QM algorithm is useful for computer program based simplification of Boolean Expression containing large number of variables.
- Comparison of different logic circuit implementation is done through cost criteria.
- Consideration of shared terms in multiple functions can lead to an overall minimized circuit with less cost.
- Glitches may occur in a circuit due to finite propagation delay for certain input combination. A circuit with a potential for glitch is considered to have hazard.
- Circuit can be made hazard-free by covering the hazard(s) by otherwise redundant additional term(s).

The slide features a blue header with navigation icons, a white main content area, and a blue footer containing logos for IIT Bombay, SWAYAM, and IIT Madras.

Before that let us have a quick recap of what we discussed in week 3. We saw that Karnaugh map is useful for simplification, but when it becomes 5- variable we need a 3D visualization and more than 4- variable, it is difficult using Karnaugh map and if we use entered variable map, we the map size reduces by a factor of 2. And QM algorithm, when we have more number of variables; QM algorithm is useful and we can generate a computer algorithm, a computer based algorithm a code for minimization. And when we look at various realization of a logic circuit, digital logic circuit we consider certain cost criteria which could be get input cost total number of literal cost or it total number of you know gate inputs plus total number of gates that are used.

So, using those gate criteria we can think of minimizing one particular function and of various options we can select one particular option. And when we have to realize multiple function, then we can see how we can use the shared terms that could be there amongst multiple functions. And during realization, when we are doing the minimization due to finite propulsion delay, there could be glitches that can occur and during the transition from one particular value to another and such potential glitch in a circuit gives rise to hazard and to cover hazard, we need to we need to include additional terms which is otherwise considered redundant. So, this is briefly what we discussed in the last week.

(Refer Slide Time: 02:22)

If-Then-Else

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$Y = \overline{A} \cdot C + A \cdot B$

Handwritten notes: $1 \cdot C$ and $0 \cdot B$ above the first term; $0 \cdot C$ and $1 \cdot B$ above the second term.

If A (i.e. A = 1)
Then, Y = B
Else, Y = C

A	Y
0	C
1	B

So, to discuss multiplexer we look at a truth table which is which is you know converted to a Karnaugh map realization which is what you see over here. This is something that we have seen before very simple ok.

The truth table is like this and what is this truth table saying? If when A is 0; when A is 0 right. So, the output here, output here follows C. You can see 0 0 C is 0; Y is 0; C is 1; Y is 1 as long as A is 0 and when A is 1 output follows B. So, this is 0; this is 0; this is 1 and this is 1; very simple truth table corresponding Karnaugh map and this is the realization and corresponding hardware realization, converted this particular logic equation is what you see over here.

Now, this one can be seen in a different way; the way we look at the you know algorithms. So, if A; if A means if A is true that is if A is equal to 1; then Y is equal to B. Is not it? If A is equal to 1 this term becomes 0 dot C which is 0 and this term becomes 1 dot B which is B. So, Y is equal to B. So, you can write this way when A is equal to 1 ok; Y is equal to B right. Else means when A is not equal to 1; the other option this is a binary you know possibility of 0 and 1 only is there for the variable A. ok.

So, when A is 0. So, what we see this is 0 1 dot C and this is 0 dot B ok. So, in that case the output is only C. So, for A is equal to 0 output is C. So, we can write the truth table in this manner also and this is corresponding you know meaning in terms of the codes that we write for algorithms ok.

(Refer Slide Time: 04:48)

Multiplexer

$Y = A'C + A.B$

Consider,
 A = Control (Select) input, S_0
 B = Data input, D_1
 C = Data input, D_0

$Y = S_0' \cdot D_0 + S_0 \cdot D_1$

S_0	Y
0	D_0
1	D_1

A multiplexer steers one of the many inputs to an output based on control input(s).

2-to-1 MUX

The diagram shows a 2-to-1 MUX block with inputs D_0 and D_1 , and a select input S_0 . The output is Y . A circuit diagram to the right shows the internal logic of the MUX, where S_0 is connected to two AND gates. The first AND gate has inputs S_0' and D_0 , and the second has inputs S_0 and D_1 . The outputs of these two AND gates are connected to an OR gate, which produces the output Y .

Now, how is it related to multiplexing operation? So for that, we define what is a multiplexer? Multiplexer steers one of the many inputs; one of the many inputs to an output based on what the control input says ok. This is what is the job of a Multiplexer. So, here there is an example where which is doing a 2 to 1 multiplexing; that means, 2 inputs are there and 1 output is there. We shall see example of you know 4-to-1 and 8-to-1 and other things.

So, this is the transmission line say and these inputs is not required to transmit every point of time. So, when at certain point of time D_0 wants to transmit; at some other point of time D_1 wants to transmit. So, there is no need of putting a dedicated channel for each of D_0 and D_1 because this channel is very long it according to the cost gets increased. So, here 1 channel can be used by D_0 and D_1 depending on the need and how it is decided, who will take control over the transmission channel that is the Y . It is decided by S_0 .

So, when S_0 is 0 D_0 you know gets connected to Y and when S_0 is equal to 1, D_1 gets connected to Y . So, that is the idea. Is it ok? So, if there are 4 such things, then there will be 4 such inputs and then, the select lines will be accordingly which we shall see later right. So, this particular thing and what we just discussed in the previous slide, you can see there is a similarity. So, here what we are looking at? In this case you are looking at a an expression which is something like this. So, S_0 is

equal to 0; Y is equal to D naught and when S naught is equal to 1, Y is equal to D 1. This is similar to this relationship ok.

So, the same circuit in which what we had seen for A, if we assign that to S naught that is the select input and B and C becomes data inputs which is given to D 1 and D naught respectively. Then the previous circuit behaves like a 2 to 1 multiplexer ok, where A is your S naught; D naught is your C and D 1 is your B ok. The same circuit, the same hardware realization that you had seen and then, we can represent it in the form of a block ok. So, this is the block within which that hardware, the hardware that we had seen before is there 2 AND gate and 1 OR gate. So, this S naught when it is 0, this is the control input D naught is actually getting connected to Y; when S naught is equal to 1 D 1 is getting connected to Y right.

So, that is the idea and this is the way a simplified block level representation is made and we can make use of this block in subsequent discussion. Knowing that inside the block there is a the hardware that we had seen before with 2 input 2 and gate and 1 or gate that is what makes it 2 to 1 multiplexer.

(Refer Slide Time: 08:14)

4-to-1 Multiplexer

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3$$

Handwritten notes: $1.1. D_0 \rightarrow 0 + 0 + 0$, $S_1=0, S_0=0$

Now, we look at 4-to-1 multiplexer. So, 4-to-1 multiplexer as per our previous understanding, there will be 4 data inputs D naught, D 1, D 2, D 3. This is the way we designate them and this is one output right and 1 of them will get access to Y depending on the selecting lines. Again, this S 1 and S naught are binary variables. So, for selecting

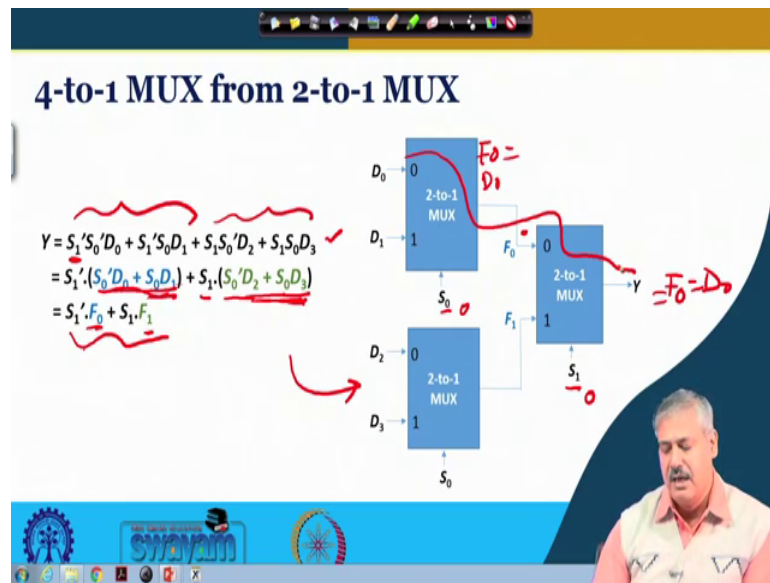
4 of them, we need 2 of them; 2 of 2 select lines. So, in S_1 and S_0 are 0 and 0, D_0 is getting selected; that means, Y is equal to D_0 right. When they are 0 1, D_1 ; when 1 0, D_2 and when 1 1, this is D_3 ok. So, how do we write it in the form of a logic equation?

So, this is the way we can write it right. So, when we substitute S_1 and S_0 as 0 and 0 both of them are 0. So, this is $0'$ that is 1. This is 1; this is D_0 and rest of the term one of S_1 or S_0 will be 0 for being 0 will generate the 0. So, plus 0 plus 0 ok. So, Y is nothing but D_0 . So, for S_1 is equal to 0 and S_0 is equal to 1. This is the term associated with D_1 which will be the value of Y and the other terms will be this term, this term, this term; this 3 term will be 0. This is how we can write the expression ok.

And the corresponding hardware is what you see over here. So, S_1 and S_0 right. So, amongst them this 4 possibilities are there. So, 3 input AND gates; these are the selections. This is D_0 , D_1 , D_2 , D_3 . So, if it is 0 0. So, this is 0; this becomes 1; this becomes 1 and this is D_0 . So, Y is equal to D_0 at that time and rest of the values these and gates at that time will be 0 only. This is D_0 ok, for S_1 and S_0 at 0. So, similarly for other values also you can take and you can see that the output is in the following this particular truth table ok.

So, 4-to-1 multiplexer, when we give a block diagram representation; so this is the block diagram representation 2 select inputs S_1 and S_0 and 4 data inputs. These are the 4 data inputs and there is 1 output and 2 select inputs. The first one is S_1 and second one designates S_0 ok. So, 0 0 0 1 1 0 and 1 1 and corresponding inputs are you know assigned here and this is getting connected to Y depending on the S_1 and S_0 value. So, what is there inside this block? This hardware is there ok. This hardware is there, this particular logic circuit is there right. So, when we use this as a block we know that inside this is the logic circuit that is there clear.

(Refer Slide Time: 11:28)



Now, let us look at the 4-to-1 multiplexer equation a bit you know closely. So, this was the basic equation, we had seen before. Now, if we take S 1 prime common between these two right. So, we get S naught time D naught, S naught D 1. And if we take S 1 common between these two; we get S 1 out and S naught time D 2 and S naught D 3 within the bracketed term. Now if you look closely at this two, this particular term, this particular two terms you know SOP term. What is it? It is nothing but a 2-to-1 multiplexer equation and what it this? This is also a 2-to-1 multiplexer.

So, basically you have got this realized by a 2-to-1 multiplexer and this is realized by another 2-to-1 multiplexer and if we write this particular function as F naught and this one as F 1 you have got S 1 time F naught and S 1 F 1 and this again [laughter] is a 2-to-1 multiplexer equation, where F naught is your D naught and F 1 is your D 1, the basic equation if you remember ok. So, essentially in the 4-to-1 multiplexer if you see right, there is a there is 1 2-to-1 multiplexer; 1 2-to-1 multiplexer and finally, another 2-to-1 multiplexer.

So, three 2-to-1 multiplexers are there. So, if we you know write it the terms of 2-to-1 multiplexer. So, this is how what we can see this particular equation getting manifested in the form of you know hardware realization inside you have got 2-to-1 multiplexer circuit, the way we have seen before ok. So, when S 1 is say 0 and S naught is 0, what is happening? S 1 0 means F naught is selected at the output and when S naught is 0, D

naught is selected as F naught ok. So, at that time then when S naught is 0, this is D naught is your F naught; F naught equal to D naught and Y is equal to F naught is nothing but D naught.

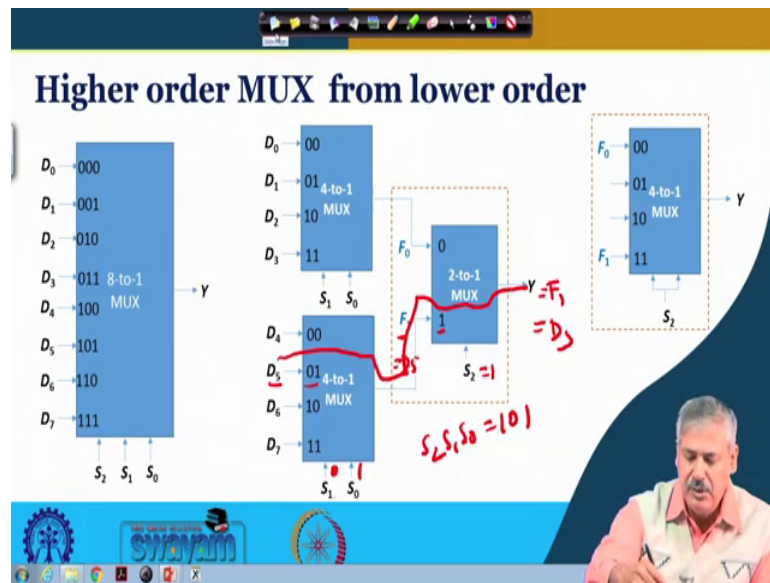
(Refer Slide Time: 13:58)

4-to-1 MUX from 2-to-1 MUX

$$\begin{aligned}
 Y &= S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3 \\
 &= S_1'(S_0'D_0 + S_0D_1) + S_1(S_0'D_2 + S_0D_3) \\
 &= S_1'F_0 + S_1F_1
 \end{aligned}$$

So, this line will be; this line will be going there and if it is instead S 1 is 1 and S naught is 0 F 1 is 1 here. This line will be selected F 1 will be selected S naught is equal to 0 means, this 1 will be selected. So, F 1 will become D 2 at that time right and Y will become F 1 equal to D 2. So, basically you are having this line going to output for S 1 is equal to 1 and S naught equal to 0. Is it clear?

(Refer Slide Time: 14:36)



So, we can see that this way higher order multiplexer can be obtained from lower order multiplexer in a; so, there is a generalized you know approach for this. So, this the earlier one was 4-to-1 multiplexer was obtained from 3 2-to-1 multiplexer. Now let us look at the example of getting 8-to-1 multiplexer from lower order multiplexers. So, 8-to-1 multiplexer how many selective inputs will be there? 8 inputs.

So, selective inputs will be with 3 selective inputs. We can get 2 to the power with 3 selective inputs, we can get 2 to the power 3 up to 8 input line getting selected and getting connected to the output steer to the output right. So, 3 inputs are 3 selective inputs are there and these are D_0 to D_7 these are the data inputs and the S_2, S_1, S_0 that is the 0 0 0. So, this is the corresponding data input D_0 is getting connected to where they are 0 0 0. So, any other value say 1 0 1 1 0 1. So, D_5 will get connected to Y . So, this is the understanding right.

And how we can get it; get 8-to-1 multiplexer from lower order. So, we can have 2 4-to-1 multiplexer. We can have 2 4-to-1 multiplexer, the similar way you have expanded the equation before you can you know go with the same exercise. So, this is 1 4-to-1 multiplexer and this is another 4-to-1 multiplexer with S_1 and S_0 right which is generating F_0 or F_1 ; F_0 and F_1 2 functions and S_2 is a 2-to-1 multiplexer which is selecting which of this F_0 or F_1 will be going to the output and F_0 and F_1 will take 1 of these input values depending on S_1 and S_0 that is there. So,

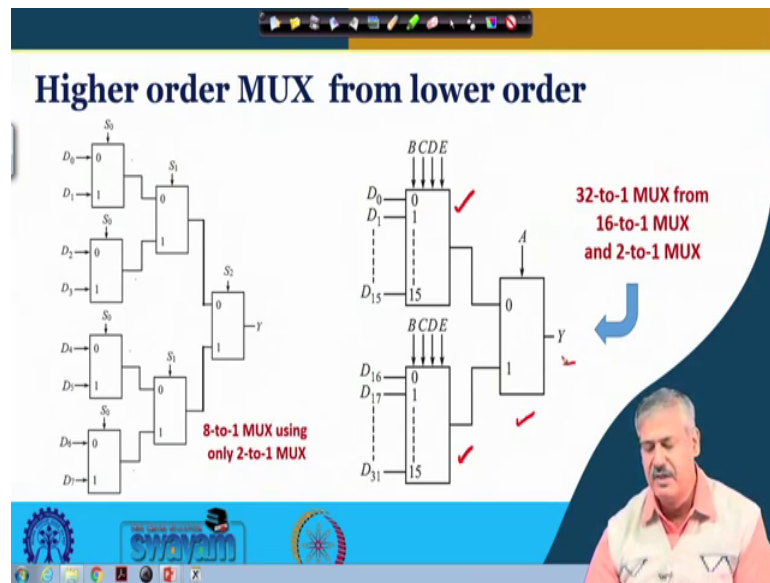
again we take the example of S_2, S_1, S_0 as 0 0 0. So, $S_2 = 0$ means F_1 will be selected right and $S_1 = 0, S_0 = 0$ means D_5 will be selected right. So, this is the path that will be there.

So, if we take an example of S_2, S_1, S_0 as 1 0 1 what will happen? $S_2 = 1$. So, basically F_1 will be selected; this 1 will be selected as going to the output right. So, $Y = F_1$ will be there and this is 0 1; $S_1 = 0$ and this is 1 ok. So, 0 1 is this is the 1; D_5 will be selected. So, F_1 will become D_5 . So, $Y = F_1 = D_5$. So, D_5 will go to the output, absolutely no issue. So, this is what you can see happening when you have got you have got higher order multiplexer realized by lower order.

Now, imagine a situation where, you have got in the store only 4-to-1 multiplexer; you do not have 2-to-1 multiplexer right. So, can we realize it? Of course, because we can always get a lower order multiplexer from a higher order. Getting higher order from lower order, we have to follow a cascaded approach stage by stage approach even that we have seen. Now to get a lower order from higher order. So, this is one example where we are getting a 2-to-1 multiplexer from a 4-to-1 multiplexer. What we are doing? So, there are 2 select inputs. It is made I have made common.

So, S_2 is connected over here. So, whenever we place 0, both the select inputs become 0 ok. So, this will be selected and when you know we place 1. So, this 2 become 1. So, 1 you know this, this one will be selected. So, either this gets selected or this gets selected right. So, either this one goes to the output or this input goes to the output. So, this is nothing but 2-to-1 multiplexer; one of this two going to the output. So, you can replace this block with this one and accordingly, you can get the a 8-to-1 multiplexer using only 4-to-1 multiplexer, otherwise 2 4-to-1 multiplexer 1 2-to-1 multiplexer is sufficient.

(Refer Slide Time: 19:19)



Now, in the previous example we had seen 2 4-to-1 multiplexer getting connected to 1 8-to-1 multiplexer to give you ultimately a 1 8-to-1 multiplexer and before that we had seen that 4-to-1 multiplexer can be obtained by 3 2-to-1 multiplexer. So, this is 1 4-to-1 multiplexer that we have seen before which we can replace with 3 2-to-1 multiplexer and this is another one which can be replaced with another 3-to-1 multiplexer. So, this is essentially a 4-to-1 multiplexer ok.

And finally, this is a 2-to-1 multiplexer. So, how ultimately how many 2-to-1 multiplexer we can use to get the 8-to-1 multiplexer 1 2 3 4 5 6 7. So, with 7 2-to-1 multiplexer, we can get 1 8-to-1 multiplexer realized ok. So, this is one thing that we can take note of that we can go up to that level. We can further break a 4-to-1 multiplier to 2-to-1 multiplexer and realize it ok.

And this is one example where a 32-to-1 multiplexer is obtained from a from 16-to-1 multiplexer and 2-to-1 multiplexer. So, 2 16-to-1 multiplexer are there, they are generating 2 output like D 1 that is they are before and 1 2-to-1 multiplexer is combining these 2 outputs and is generating an output. So, these are different examples by which I hope it is clear from to you that how higher order multiplexer can be obtained from lower order and also we have seen how lower order can be obtained from higher order ok.

(Refer Slide Time: 21:14)

Shanon's Expansion Theorem and MUX

$$F(A,B,C) = A'B'C'D_0 + A'B'C'D_1 + A'BC'D_2 + A'BC'D_3 + AB'C'D_4 + AB'C'D_5 + ABC'D_6 + ABC'D_7$$

A	B	C	F(A,B,C)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
...

$D_0 = 1$
 $D_1 = 0$
 $D_2 = 0$
 $D_3 = 1$

Now, we shall look into Shanon's Expansion Theorem and it is use in the multiplexer you know design.

(Refer Slide Time: 21:19)

Shanon's Expansion Theorem and MUX

Shanon's Expansion Theorem: (inherent If-Then-Else)

$$F(x_1, x_2, x_3, \dots, x_N) = x_1' \cdot F(0, x_2, x_3, \dots, x_N) + x_1 \cdot F(1, x_2, x_3, \dots, x_N)$$

$$Y = S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3$$

$$= S_1' \cdot [0'S_0'D_0 + 0'S_0D_1 + 0.S_0'D_2 + 0.S_0D_3] + S_1 \cdot [1'S_0'D_0 + 1'S_0D_1 + 1.S_0'D_2 + 1.S_0D_3]$$

$$= S_1' \cdot (S_0'D_0 + S_0D_1) + S_1 \cdot (S_0'D_2 + S_0D_3)$$

$$F(x_1, x_2, x_3, \dots, x_N) = x_1' \cdot [x_2' \cdot F(0, 0, x_3, \dots, x_N) + x_2 \cdot F(0, 1, x_3, \dots, x_N)] + x_1 \cdot [x_2' \cdot F(1, 0, x_3, \dots, x_N) + x_2 \cdot F(1, 1, x_3, \dots, x_N)]$$

So, Shanon's Expansion Theorem, we this is we are revisiting it we have seen it before ok. So, Shanon's expansion theorem, this is the one remember this one. So, given any function F X 1 this is the number of variables, I mean it could be 3 variable for variable to variable, I mean just it is generalized expression. So, X 1 prime so, corresponding term X, wherever X 1 is there it becomes 0 ANDed with the function X 1 prime replaced

by 0 plus or'd with X_1 and the function wherever X_1 is there, you replace it with 1 whatever comes up. So, you just simply sum these 2 particular products, you get the original function.

So, this is Shannon's expansion theorem which you have seen before. Now in the 2-to-1 multiplexer, you know context we can see if this is my X_1 and this is select input for 0, one will go; for 1 the other will go as output. So, if this is my Y ; this is Y , then if X_1 is 0. This is the term that is going. This is my corresponding D_0 equivalent to D_0 and when X_1 is equal to 1, this one will go to the output because X_1 is equal to 1 means this term will become 0; the top term will become 0 ok. So, the output will become this particular expression ok, which was given before.

So, there is an inherent if then else, if X_1 then X_1, X_2, X_3 to X_n . This particular function and if not X_1 , else that is then F_0, X_2, X_3 to X_n . Then this particular expression will be the function output. This is similar to the multiplexing action that we see and this we can look at, we can put it into the basic multiplexers 4-to-1 multiplexer equation and for S_1 taken out as X_1 wherever S_1 is appearing we can put 0 0 0 0 and the other one S_1 ; this was S_1 prime wherever S_1 is there, we put 1 1 1 1 and then, we simplify we see that we get back to what we had seen before. So, this is 1 2-to-1 multiplexer equation giving F_{naught} and this is another 2-to-1 multiplexer equation giving F_1 and finally, these 2 are getting combined this is your the third 2-to-1 multiplexer.

So, similar and exactly similar to what we had seen earlier and we also note that the Shannon's expansion theorem can be nested. So, first you take X_1 out. So, whatever you get this specific this specific equation you can take X_2 out right. So, by which you can write this expression and the other expression and similarly, you can look at this expression. This expression sixth one you can take X_2 out X_2 prime here and X_2 over here. So, this one wherever X_2 appears. In this case one X_2 prime is taken out.

So, 0 will be there when X_2 is there X_1 will be there. This is the nested version of the Shannon's Expansion Theorem which you have seen before and so, this is equally applicable over here right and if you do that; if you do that the earlier 8-to-1 multiplexer realization the one that we had seen using 2-to-1 multiplexers right, by nesting we can see is shown in this particular diagram. So, this is your $F(A, B, C)$ right, the basic

equation, A is your select input. So, whenever A is 0 right, this one is going as the input and whenever A is 1, this one is the input right.

Now, this $F_0(B, C)$ this is the output of another 2-to-1 multiplexer. So, whenever this is the B wherever B the first one whenever B is equal to 0, we put $F_0(0, C)$ will be the input and whenever A B is equal to 1 $F_0(1, C)$ will be the input ok. Similarly there will be the other line. And finally, this $F_0(0, C)$, if C is taken as the select input out. So, this 0 0 terms are there so, C is equal to 0 is this term and C is equal to 1 is this term. So, accordingly we have this 8 possibilities; these 8 possibilities $F_0(0, 0)$ to $F_0(1, 1)$ is over here and the individual output that are generated at every level this follows the Shannon's expansion theorem, the nested version which we had seen in each of the cases and this we can see over here also right and for any value A B C we have already noted this particular term.

So, if we have a function that we want to realize using this particular you know multiplexing architecture. So, this is A B C and this is say this is the corresponding truth table. So, wherever there is a 1 in the truth table right; so, if A B C F 0 0 0; so, this is $F_0(0, 0)$ right. So, if there is a 1; so there is a 1 over here. So, we put the corresponding D naught value as 1 0 0 1, so this is the corresponding term we see the truth table out in this particular place is 0. So, this is my corresponding input ok so, we put D 1 as 0 ok.

So, this shows that using multiplexer, we can realize the truth table. So, inside this is what this whole block this whole block is a 8-to-1 multiplexer. We have realized it in this manner, but effectively this is a 8-to-1 multiplexer. So, if input to that is $F_0(0, 0)$ to $F_0(1, 1)$ over here ok. This can map to the truth table by appropriately placing the values of D 1, D 2, D 3 upto D 7 ok; making them zeros and ones as per the truth table (Refer Time: 28:20) we have seen it. More of this, we will see in the next class of the multiplexer.

(Refer Slide Time: 28:31)

Conclusion:

- A multiplexer steers one of the many inputs to an output based on control input(s).
- n control inputs can select up to 2^n data inputs and steer it towards output.
- Higher order multiplexer can be obtained from lower order by cascading.
- Lower order multiplexer can be obtained from higher order by appropriate connection of select inputs.
- Shannon's expansion theorem and its inherent if-then-else structure is useful in getting insights of multiplexer operation.

So, to summarize a multiplexer steers one of the many inputs to an output based on control inputs, n control inputs can select up to 2^n data inputs and steer it towards the output. Higher order multiplexer can be obtained from lower order by cascading. Lower order multiplexer can be obtained from higher order by appropriate connection of select inputs. Shannon's expansion theorem and its inherent if-then-else structure is useful in getting insights of multiplexer operation.

Thank you.