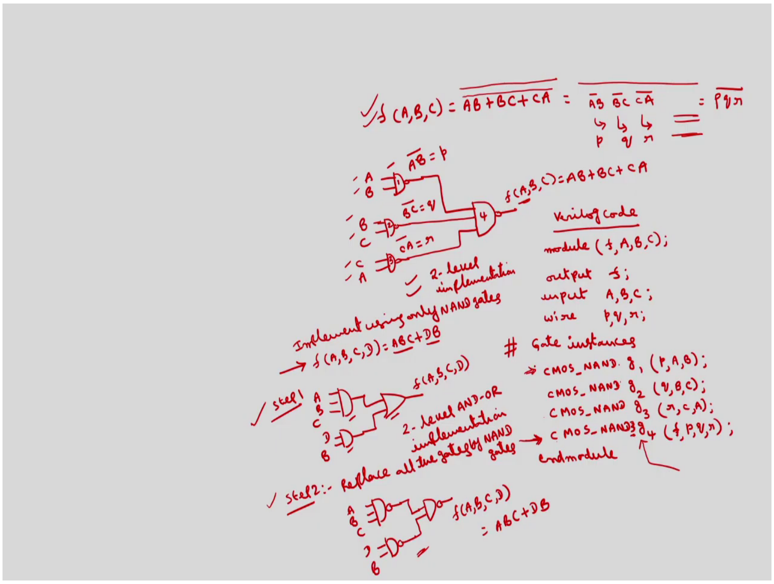
System Design Through VERILOG

Prof. Shaik Rafi Ahmed
Department of Electrical and Electronics Engineering
Indian Institute of Technology, Guwahati

Switch level modeling
Lecture - 12
Modeling using transmission gates, CMOS delay times

In the last class, we have discussed about the implementation of a Boolean function.

(Refer Slide Time: 00:39)


$f(A,B,C) = \overline{\overline{AB+BC+CA}} = \overline{\overline{AB}\,\overline{BC}\,\overline{CA}} = \overline{pqr}$
$\overline{AB} = p$
$\overline{BC} = q$
$\overline{CA} = r$
$f(A,B,C) = AB+BC+CA$
2-level implementation
Implement using only NAND gates
$f(A,B,C,D) = ABC + DB$
$f(A,B,C,D)$
step1
2-level AND-OR implementation
step2:- Replace all the gates by NAND gates
$f(A,B,C,D) = ABC + DB$
Verilog code

```
module (f, A, B, C);
output f;
input A, B, C;
wire p, q, r;
# Gate instances
CMOS_NAND g1 (p, A, B);
CMOS_NAND g2 (q, B, C);
CMOS_NAND g3 (r, C, A);
CMOS_NAND3 g4 (f, p, q, r);
endmodule
```

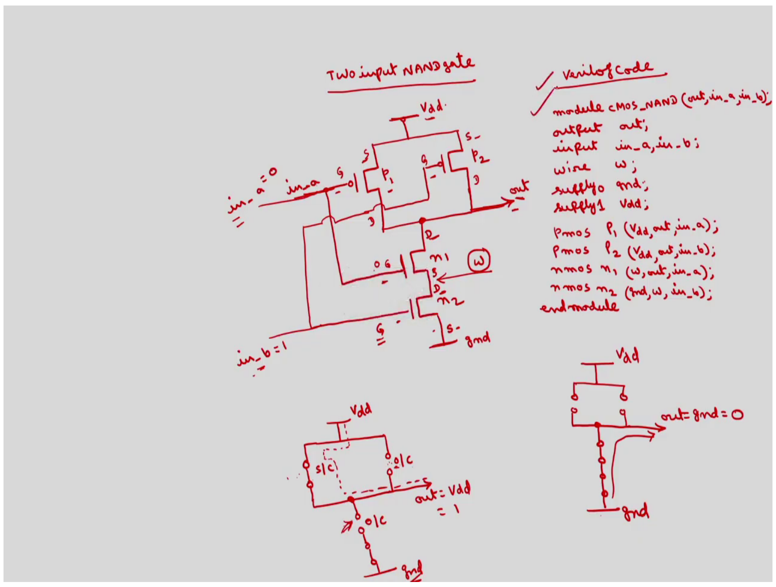
f of A, B, C is equal to AB plus BC plus CA using the CMOS gates. So, there is another way to implement this Boolean function by instantiating the NAND gates. As I have discussed in the earlier lecture also this f of A, B, C can be expressed in the form of double complement.

The double complement will give the same value. So, if I evaluate the inner complement using De-Morgan's law this will be AB bar BC bar CA bar whole bar. Now, I found implement this function using the NAND gates. So, we require one 2-input NAND gate. So, this will give AB bar. Let us call this AB bar as some p, this as some q, this as some r. So, this is nothing but now pqr whole bar, NAND operation of pqr.

So, p is nothing but this AB bar, this is BC bar which is q and CA bar is r. So, these two we have to NAND again. So, this will give f of A, B, C is equal to this AB plus BC plus CA. So, we have directly implemented in the last lecture. Now, we can instantiate the NAND operation. Here how many NAND gates are there? Total we have 4 NAND gates if I call this gate as 1 2 3 4, 4 times if I instantiate this NAND operation we can obtain the Verilog code of this f of A, B, C is equal to AB plus BC plus CA.

So, if I write the Verilog code for this one. So, overall this will be having one output I will call as f and three inputs AB C module f comma A, B, C; output f; input A, B, C then in addition to this we require some wires. Let us recall this one as p, this we call as q, this we call as r. So, I will write this pqr as wires. Then we have to instantiate gates. So, we can call back the NAND program that we have written in the earlier class.

(Refer Slide Time: 04:31)



Verilog code

```
module CMOS_NAND (out,in_a,in_b);
output  out;
input   in_a,in_b;
wire    w;
supply0 gnd;
supply1 Vdd;
pmos  P1 (Vdd,out,in_a);
pmos  P2 (Vdd,out,in_b);
nmos  n1 (w,out,in_a);
nmos  n2 (gnd,w,in_b);
endmodule
```

This is the NAND logic that we have written. The name that they have given as CMOS underscore NAND, then we have to specify out in A in B. So, now, if I call the same name CMOS underscore NAND, CMOS underscore NAND I am calling this gate as gate 1 say g1. So, we have to give the outputs and inputs of these; output is p, inputs are A and B.

Similarly, CMOS underscore NAND g2, output is q, inputs are B and C. Similarly, CMOS underscore NAND g3, output is r, inputs are C and A. Then finally, another CMOS NAND g4, the output is final output f of A, B, C, inputs are p, q and r. This you call as otherwise

CMOS NAND as it is, this you call as CMOS NAND 3 because you have written this program for CMOS 2-input NAND gate.

So, in a similar way you can write for 3-input NAND gate also. So, in this case we have to call a 3-input NAND gate assuming that there is a 3-input NAND gate Verilog code is there that name if I call this one as CMOS NAND underscore 3, that is why I called as 3. These three are same because these are 2-input NAND gates whereas, this is 3-input NAND gate I will assume that there is a Verilog code similar to these two input NAND gate for 3-input NAND gate also whose name called as CMOS underscore NAND 3.

Then this one. I cannot use this CMOS NAND here because this is 3 input NAND gate, but the code I have written is for 2 input NAND gate. So, here I am assuming that for 3-input NAND gate also there is a code in a similar manner you can write. So, this is end module. This is how we can instantiate the NAND operation to implement any Boolean function.

In general, this particular implementation is, you can write in terms of the NAND operations, you can draw the 2 level AND-OR operation and then we can replace all the gates with NAND gates. This is valid for only 2-level implementation. So, f of A, B, C, D is equal to ABC plus DB, say.

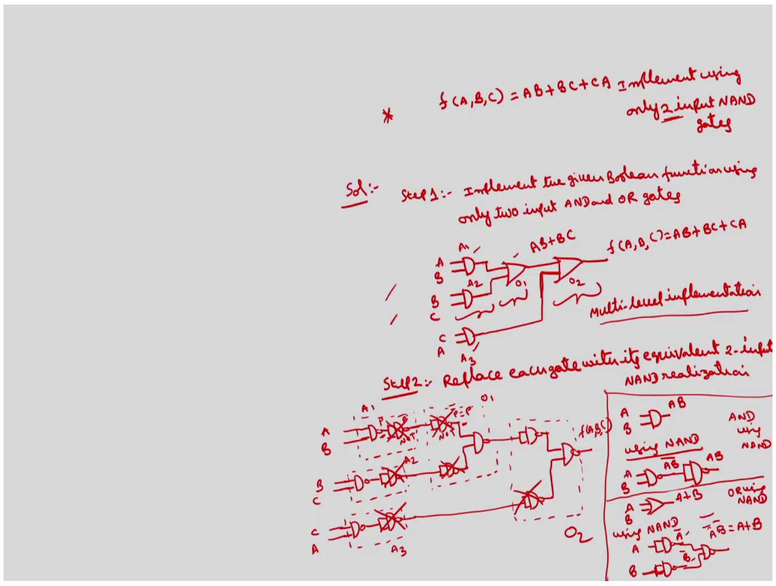
So, if you want to write this one, you first write 2-level AND-OR operation; AND operation is ABC, this is AND; this is another AND DB, this is ABC then we require a OR gate to obtain this f of A, B, C, D. This is called 2-level AND-OR implementation. So, you replace in order to implement any Boolean function using only NAND gates. So, this is the procedure, this is step 1.

Step 2 is replace all the gates by NAND gates. So, you replace this by NAND, this by NAND, even OR by NAND. We will get the NAND implementation. This is ABC, DB; this is f of A, B, C, D is equal to ABC plus DB. This is how we can implement the NAND implementation.

This of course, we are going to obtain using this De-Morgan’s law only, but in general without using the De-Morgan’s law also, this is two step procedure to implement any Boolean function using the NAND gates.

First we have to draw the 2-level AND-OR implementation and then we have to replace all the gates by NAND gates. Suppose here if I want to use only two input NAND gate because in most of the cases we will be having only two input NAND gates. So, given the same function, how to implement using only two input NAND gates? So, I have two input NAND gate Verilog code which is CMOS underscore NAND. So, for that also there is a procedure.

(Refer Slide Time: 10:43)



f(A,B,C) = AB+BC+CA Implement using only 2-input NAND gates
Sol:- Step 1:- Implement the given Boolean function using only two input AND and OR gates
AB+BC
f(A,B,C)=AB+BC+CA
Multi-level implementation
Step 2:- Replace each gate with its equivalent 2-input NAND realization
AND using NAND
OR using NAND
A+B
AB = A+B

If we want to implement the same Boolean function f of A, B, C is equal to AB plus BC plus a CA. So, in earlier example I have implemented this using three 2-input NAND gates and one 3-input NAND gate. Suppose, if I want to implement using only 2-input NAND gate. If there is no restriction on the number of inputs if I want to use any number of inputs for the gate, then the procedure that I have explained in the previous slide is we have 2-step procedure.

Step 1 is we have to draw the 2-level implementation and then you have to replace all the gates by NAND gates. If I want to implement using only 2-input NAND gates then, here also we have to draw these 2-level or the 2-input AND-OR implementation.

So, step 1 here is implement the given Boolean function using only 2-input AND and OR gates. So, what is the implementation of this one? AB, this is AND operation, this is BC, another AND operation. So, I want to use only 2-input OR gate. So, I have to first obtain this

AB plus BC using this two input OR gate, then you obtain CA, then you take another 2-input OR gate.

So, this is your f of A, B, C is equal to AB plus BC plus CA. This is no more 2-level. So, this is one level, from here to here one level, here to here another level, here to here another level. This is 3-level implementation or you can generally called as multi-level implementation.

So, in the previous example we have taken only 2 level. Now, if I want to implement by using any number of inputs for the NAND gate, we have to draw first 2 level, but if I want to use only two input NAND gates then the first step is we have to use multi level implementation where we can have only two inputs for any gate, AND gate or OR gate.

Then step 2 is you have to replace each gate with its equivalent 2-input NAND realization. So, we know that a NAND gate can be used to implement all the gates because the NAND gate is universal gate. So, AND operation, how to implement the AND gate for example, if you have this A, B, this is AB I want.

So, using NAND how to implement this? Means simply if I take the NAND, AB we will get AB bar, but we want AB only. So, what we have to do? We have to take a NAND gate, but the inputs are same. AB bar, AB bar whole bar will be AB double bar which is equal to AB.

Similarly, if I want OR; suppose if you have something like AB only OR, I want A plus B. So, this is using NAND using NAND. So, first you write A, this two inputs are same; A into A whole bar is equal to A bar. This is another B, B bar then you take another NAND gate. This is A bar into B bar whole bar. This is A bar, this is B bar, A bar B bar whole bar which is equal to A plus B according to De-Morgan’s law.

So, this is how we can implement AND using NAND and this is OR using NAND. You have to do this equivalents here. Now, what will be this? This NAND is nothing but NAND followed by a NOT gate this is in fact, NOT gate. This is AB. This entire thing is if I call this one as A1 gate AND 1. So, this entire thing is A1. Similarly, if I call this gate as A2, this gate as A3, this gate as O1, this gate as O2.

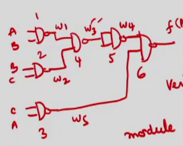
So, what is A2? A2 is similarly this, this is BC, this is A2 and this is A3 then A1, A2 outputs we have to apply to the OR gate O1 which is nothing, but two NOT gates in series then another NAND gate. So, this entire thing is O1, this is O1 and O2 is one of this output is this, this is O1, this is another input for O2, finally another NAND gate. This is your f of A, B, C. This entire thing is your O2. This is the complete NAND equivalent of the given circuit.

Now, here we have further simplification. In step 3, if two NOT gates, this is actually in fact, NOT gate; this is NOT, this is NOT. If two NOT gates appear in the series we have to cancel because if the signal is here P, this is P bar, again here P double bar which is equal to P means this point and this point are same. So, these two are useless. We can remove these.

Similarly, these two we can remove because these two are in series. Similarly, these two we can remove. So, this we cannot remove, remaining how many two input NAND gates will be there? 1 2 3 4 5 6. So, this is the equivalent circuit of given f of A, B, C is equal to AB plus BC plus CA using only two input NAND gates.

(Refer Slide Time: 19:37)

step3:- Cancel all $\overset{\text{Two}}{\times}$ NOT gates in series.



$f(A,B,C) = \underline{AB}+BC+CA$

Verilog code (using only two input NAND gates)

```
module  f_ABC (f, A,B,C);

output  f;
input   A,B,C;
wire    w1,w2,w3,w4;
# Gate instances #
CMOS_NAND g1 (w1,A,B);
  ''      g2 (w2,B,C);
  ''      g3 (w5,C,A);
  ''      g4 (w4,w3,w2);
  ''      g5 (f, w4,w5);
endmodule
```

So, the last step is two NOT gates in series, then the equivalent circuit will be. So, here we have three gates. Then finally, we have one two gate, here one gate, one gate. So, you see the equivalent circuit. This is NAND, this is NAND, here we have another NAND and here we have another two input NAND of course this is NOT gate, this is NAND. Finally, we have NAND. This is f of A, B, C is equal to AB plus BC plus CA. This is AB, BC, CA right. This is the equivalent circuit 1 2 3 4 5 6.

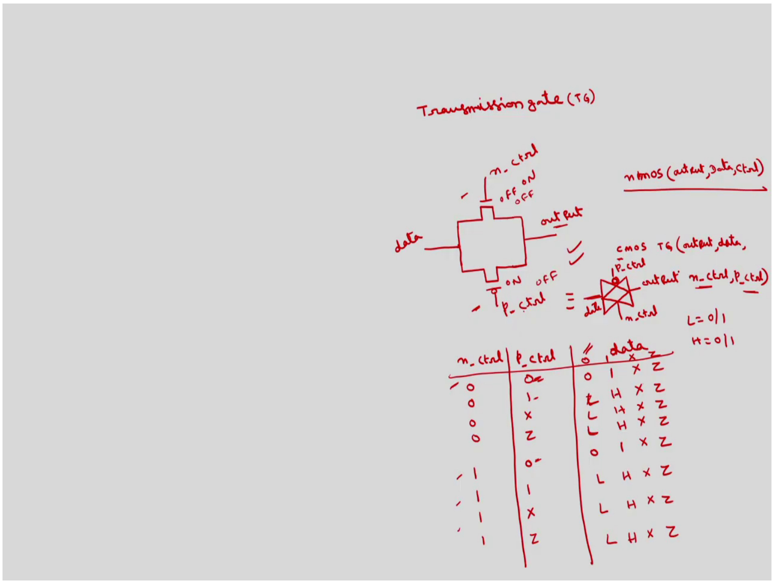
Now, I can use the same CMOS NAND the Verilog code which you have written for the 2-input NAND gate to implement the same function without writing a three input NAND gate Verilog code. Now, what is a Verilog code using only two input NAND gates? Module you can give some name. You can call this as f of A, B, C.

So, in the previous code also, I can give some name, this is but optional. So, output is f, inputs are A, B, C, output f input A, B, C, but here also I require some wires. Let us call this one as W1, W2, W3, W4, W5, total five wires are required. Then we have to use gate instances. The name that I have given for two input NAND gate is CMOS underscore NAND You give the names 1 2 3 4 5 6, I call as g1, g2, g3 so on up to g6.

g1, what is the output and what are the inputs? W1 is the output, inputs are A and B; g2, W2 is the output, B and C are inputs; g3, W5 is the output, C and A are inputs; g4, output is W4, both the inputs are W3. You see here both the inputs are W3; g5 output is f inputs are W4 and W5 and end module. This is another way to implement the given function f of A, B, C is equal to AB plus B C plus CA using only two input NAND gates.

So, total I have given three types of the implementation, one is using a series and parallel gates, another is using any number of NAND gates, third one using only two input NAND gates. So, in this way you can implement any Boolean function in fact, any digital circuit using this semi-NAND gate circuits, only two input NAND gates also you can implement. Last class you have discussed about this CMOS inverter or CMOS NOT gate.

(Refer Slide Time: 24:33)


Transmission gate (TG)

nmos (output, data, ctrl)

cmos TG (output, data, n_ctrl, p_ctrl)

L = 0/1

H = 0/1

n_ctrl	p_ctrl	0	1	x	z
0	0	0	1	x	z
0	1	L	H	x	z
0	x	L	H	x	z
0	z	L	H	x	z
1	0	0	1	x	z
1	1	L	H	x	z
1	x	L	H	x	z
1	z	L	H	x	z

So, we have another gate called transmission gate. The primitive for this one is also CMOS only. So, the transmission gate circuit will be something like. This will be input, we can also call it as data in input, there will be output. There will be two transistors, one is NMOS and other is PMOS. This is n-control, this is p-control.

So, the primitive of the key word for this one is you can write CMOS, you can give the name as TG; this is Transmission Gate, then you have to mention the output comma data which is input also of course, then you have to specify n-control comma p-control. So, ordinary NMOS and PMOS we are going to write NMOS, the general form of this one is output comma data control. This is for NMOS and PMOS, we will write PMOS whereas, for transmission gate we have to write CMOS because this consisting of both NMOS and PMOS.

So, you have to write like this. So, these n-control will be for n-MOSFET and this p-control will be for p-MOSFET. So, what is the relation between the output and input? So, in terms of n-control and p-control. So, if I write n-control, p-control, then data. What will be the output for different combinations? Totally we will have 16 combinations.

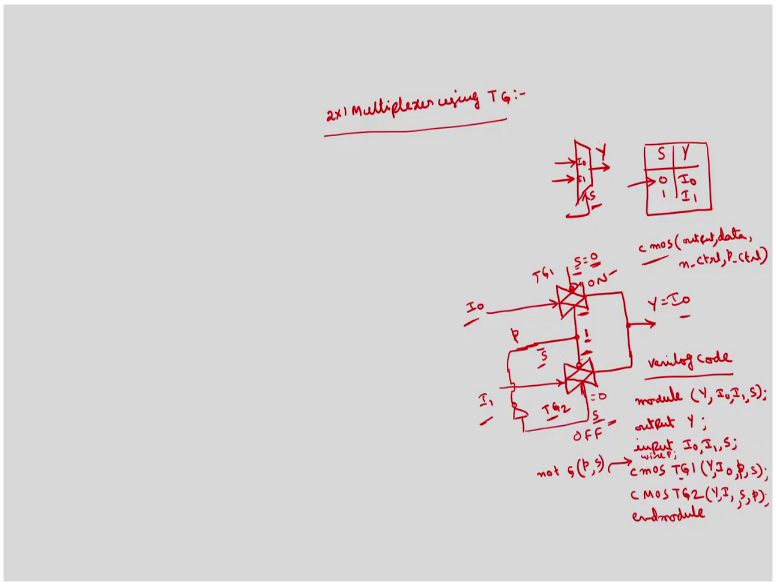
So, n-control you can fix at 0 0 0. So, this will vary from 0 1 X Z. Similarly, you can fix this at 1 1 1 1, this can be 0 1 X Z. So, corresponding data and output; if data is say for example, 0 1 X Z say if data is 0, n-control is 0, p-control is 0. So, if n-control is 0, this will be OFF, if p-control is 0, this will be ON. So, output will be input data.

So, if data is 0 0, if data is 1 1, if data X X, data is Z Z. So, this will be the case for n is equal to 0. If p is equal to 1 this is also OFF, n is equal to 0 this is also OFF. So, we will get this high impedance state, if data is 0, low; high 1 H; X Z. Here L stands for 0 by 1, H stands for 0 by 1. This I have already discussed in the earlier classes.

So, similarly if X also the same thing L H X Z, L H X Z and on the other hand if control signal is 1, n-control is 1 in all the cases this is ON whereas, this if it is 0, this will be ON. So, both will be ON means this will be 0; if 0 is 0, 0; 1 X Z. So, if both are 1s then this will be OFF, this will be ON. So, data will follow L H X Z.

Similarly, here also L H X Z. This is how we can explain the operation of this transmission gate. So, this transmission gate is again another type of this CMOS circuit which can be used to implement the Boolean functions.

(Refer Slide Time: 29:49)


2x1 multiplexer using TG:-

S	Y
0	I_0
1	I_1

cmos(output,data, n_ctrl,p_ctrl)

verilog code

```
module (Y, I0,I1,S);
output Y;
input I0,I1,S;
wire P;
not g(P,S)
cmos TG1 (Y,I0,P,S);
CMOS TG2(Y,I,S,P);
endmodule
```

If I take a simple implementation of the multiplexer using transmission gate, I will take say 2 by 1 multiplexer. So, we know that 2 by 1 multiplexer, we have 2 inputs I0, I1, one selection signal S1, output Y. So, the operation is S, output Y; S is 0, output is I0; 1 output is I1. This is the truth table of 2 by 1 multiplexer.

In order to implement by using this transmission gate, so, the transmission gate this can be equivalently written as, instead of writing these two gates and all there is a symbol used for this one is this. This will be represented by p-mos. This is p-control, this is n-control and this is data, this is output.

So, using this symbol so, how to implement this 2 by 1 multiplexer is, so, we require two such transmission gates. This is n-control, this is p-control. So, these two will be connected to this S bar, the complement of this. So, these two outputs will be connected together to this output Y, this output Y the inputs are this is I0, this is I1. These two will be connected to S, this S directly here this is complement of this.

We can easily see that the operation is when S is equal to 0 means this is equal to 1. So, for this n-type is 0, p-type is 1 this will be ON whereas, for this p-type is 1, n-type is 0. So, this will be OFF. So, ON means output Y is equal to I0.

Similarly, when S is equal to 1, this will be OFF, this will be ON, output is equal to I1. So, this is the operation of this one, then how to write Verilog code for this 2 by 1 multiplexer using transmission gates?

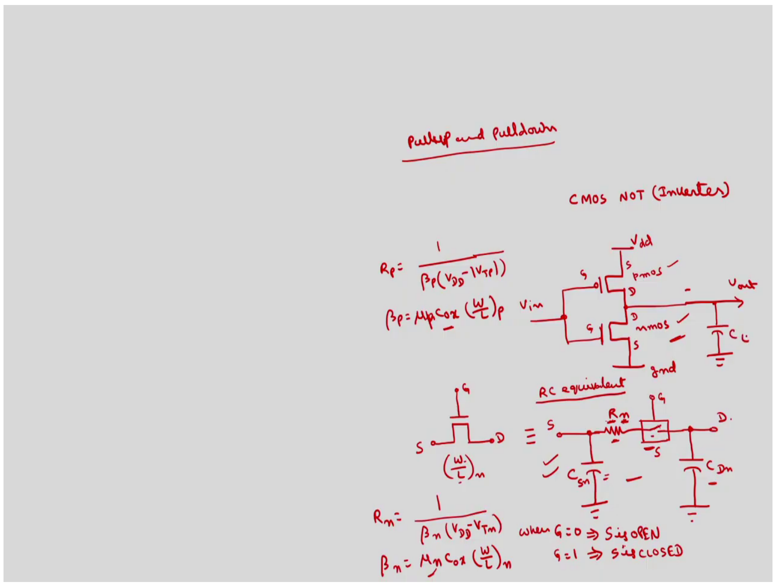
Module, output is Y, inputs are I0 I1 S, then CMOS the general primitive for this transmission gate is we call as TG1, this we call as TG2. CMOS the general structure is output data then you have to first mention n-control, then p-control. So, for TG1 output is output Y only, data is I0. What is n-control? n-control is connected to S bar.

So, we have to implement this S bar from the S using NOT operation. So, here I will write that one. So, inside this between S and S bar, there is a NOT gate that I have not mentioned, but here I will write that one is NOT, we can call as some gate G, output of the NOT gate is S bar, input is S.

So, we have S also this is S is n-control and p-control is, this is S bar we can call this S bar as some, instead of S bar we will give some name or else I will show this between S and S bar we have a NOT gate. This S we have a NOT gate. So, this also we have to define as a wire which we can call as some p say. So, p also we have defined as a wire. We can write here for n-control is S bar which is equal to p, p-control is S.

What is this p I have obtained here? This is p, S. NOT gate output is p which is S bar. Then CMOS TG1 or TG2 now output is Y only, input is I1 and what is n-control? n control is this; n-control is S itself and p-control is S bar which is p end module. This is how we can implement a 2 by 1 multiplexer using the transmission gates and the transmission gate primitives can be written in terms of CMOS using CMOS out data n-control and p-control.

(Refer Slide Time: 37:05)



Pullup and Pulldown
CMOS NOT (Inverter)
$R_p = \frac{1}{\beta_p(V_{DD} - |V_{Tp}|)}$
$\beta_p = \mu_p C_{ox} \left(\frac{W}{L}\right)_p$
Vdd
S
pmos
G
D
Vin
Vout
D
nmos
G
S
C_L
gnd
G
S
D
$\left(\frac{W}{L}\right)_n$
RC equivalent
S
R_n
G
S
D.
C_{Sn}
C_{Dn}
$R_n = \frac{1}{\beta_n(V_{DD} - V_{Tn})}$
$\beta_n = \mu_n C_{ox} \left(\frac{W}{L}\right)_n$
when G = 0 ⇒ S is OPEN
G = 1 ⇒ S is CLOSED

So, there are two more primitives which are called as pull up and pull down. In order to explain this pull up and pull down, we should know the switching characteristics of the CMOS gates. So, I will consider a simple NOT gate. If I take the CMOS NOT gate which we have discussed in the earlier classes or inverter. This is Vdd, this is ground, this is PMOS, this is NMOS, the input is connected common.

This is V in and here V out is taken. This is source, drain, gate and this is drain, gate, source, this is NMOS, this is PMOS. In order to explain the NAND and OR operation, in the last lecture I have taken the ideal case where if a transistor is ON, I have taken as a short circuit, OFF open circuit. But, in fact, if I take the practical case where there will be some resistances. So, if I take this simple NMOS FET source drain gate, if the W by L ratio width by the channel length of this NMOS is W by L n.

So, this is width of the channel, this is length of the channel. So, this is equivalent to this is source point, this is drain, this is gate, source, then followed by a resistance, then there is a switch which will be controlled by gate then drain and here we have some capacitance and here also we will have some capacitance.

This is RC equivalent of this. In most of the analysis, this type of RC equivalent model will be used this is source drain. So, this is source capacitance, this is called drain capacitance switch. So, when gate is equal to 0, this switch will be opened. Switch if I call as S; S is OPEN, OPEN means we will get just open circuit. So, no resistance will come, no capacitance and this when G is equal to 1, S is CLOSED means you will get some resistance and capacitances.

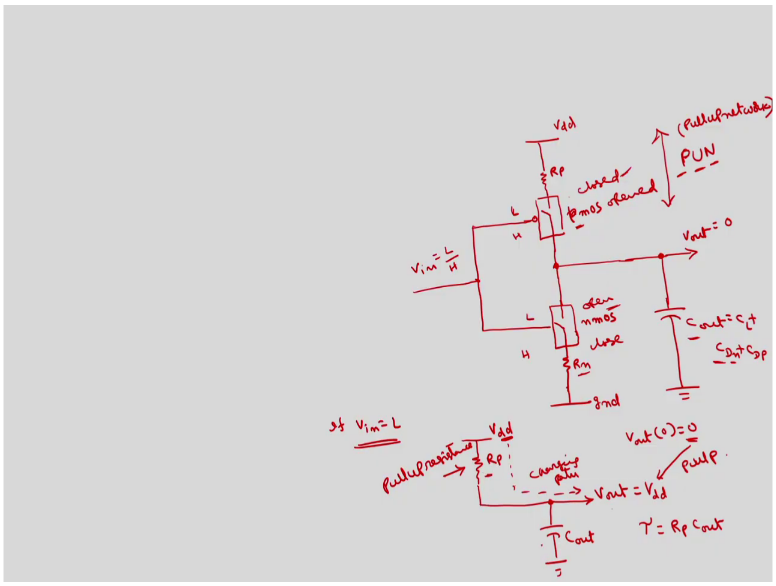
So, if I replace this by here NMOS and PMOS. We can write in a similar manner for this PMOS also. Here the resistance Rn of this the ON resistance you can call this as ON resistance, in OFF this will be open circuited. So, no resistance will come into picture that is almost infinite. So, in case of ON, this switch will be closed. So, that time the ON resistance is Rn.

This Rn expression is given by 1 by beta n into VDD minus V Tn where V Tn is the threshold voltage of the n-type MOSFET and beta n is given by mu n, C ox, W by L n. This is the equivalent circuit of this NMOS. Similarly, you can write for the PMOS only thing is here we will get Rp and this also we can call as this Sn or Dn.

So, that this is NMOS in case of PMOS this will be Sp Dp and this will be Rp. So, the relation for the Rp is 1 by beta p V DD minus modulus of V TP because for NMOS the threshold voltage is negative we will take the modulus and beta p is same mu n, C ox, W by L p.

Here mu n is the mobility of the holes here and here is the mobility of the electrons, C ox is the oxide capacitance and if I see the MOSFET this structure. I am not discussing that. So, you might have studied at semiconductor device course. So, the oxide capacitance is the C ox; W by L ratio is width by L ratio, it is the length of the channel. This is the equivalent.

(Refer Slide Time: 43:13)



If I replace this by this NMOS and PMOS, the equivalent circuit will be this is V DD, this is Rp, this gate is controlled by this one. There is another switch with Rn, this is connected to ground and this is V DD. So, the gate is connected here input and here output is taken, this capacitance here in switching analysis we have to take the capacitance also this is called C out. So, in the previous example here also we have to connect the capacitance at the output, this is V out and this capacitance is C L.

Now, this C out is equal to C L plus C Dn plus C Dp. The drain capacitance of NMOS this is PMOS, this is NMOS. So, this C out is C L plus C Dp plus C Dn. Now, depending upon whether the input is high or low, this input can be low or high. So, finally, this switch will be opened or closed. If low is applied here, this switch will be closed; if high is there, switch is opened whereas, here low will open the switch, high will close the switch.

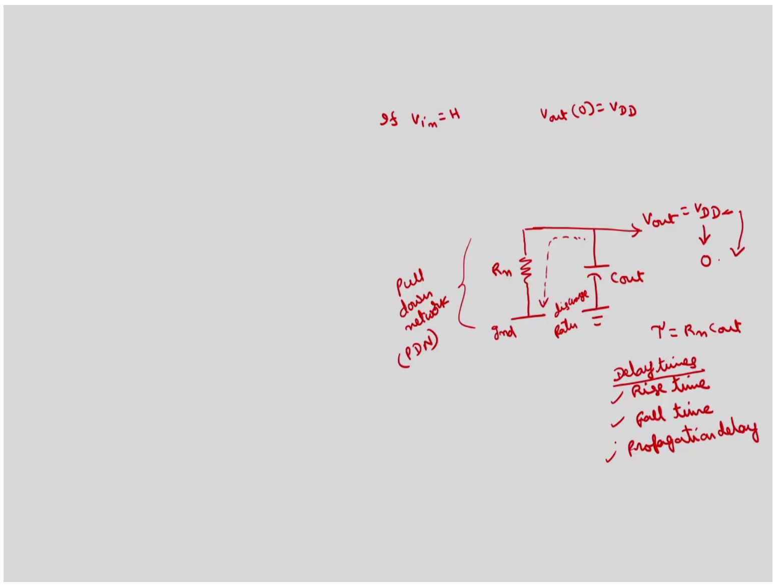
Now, if input is equal to V in is equal to low, what is the equivalent circuit? So, low means this will be closed, this will be opened, opened means this resistance will not come into picture. This is something like this one and here the open circuit only this part will be present.

This is C out, this is V out, this is R p, this is V DD. So, this lower portion this will be open circuited. So, I am not showing this that is wires.

So, what happens is now if I assume that initially V output is equal to 0 initially. V out at 0th instant is equal to 0. If I apply input is equal to low then what happens? This V out will charge to V dd. So, this is the path in which this V out will charge to V DD, this field charges slowly to V dd. So, what is the time constant at which it charges is time constant tau is equal to Rp into C out.

So, this actually the output was initially 0, now we have changed it to be V dd, this is called this output is pull up. So, that is why this particular resistance is called pull up resistance. In general, the upper network here, this upper network here above this output is called normally pull up network. So, on the other hand, if V in is equal to high, reverse case this will not come into picture, this will come ok.

(Refer Slide Time: 48:17)



Say output is initially V out 0 at 0th instant is say V DD. So, now, what will be equivalent circuit? The upper part will not be there, this is the output, V out and here we have the capacitance, C out. Here this NMOS switch will be closed, this will be connected to ground, this is R n.

So, this V out was initially V DD. Now, this will comes to 0 this is the discharging path through this because the output changes down from V DD to 0. This is why this particular network is called pull down network and rate at which it will changes from V DD to 0 is with the time constant tau is equal to R n C out.

So, using these equivalent circuits we can also derive the expressions for the propagation delay, rise time, fall time because in some cases using the CMOS primitives, we have to mention the propagation delays of the NMOS and PMOS devices also. So, for that we should know what is rise time, what is fall time, what is propagation delay, these are called delay times. So, in order to derive this delay times we have to know what is fall time, rise time and propagation delay, this we will discuss in the next lecture.

Thank you.