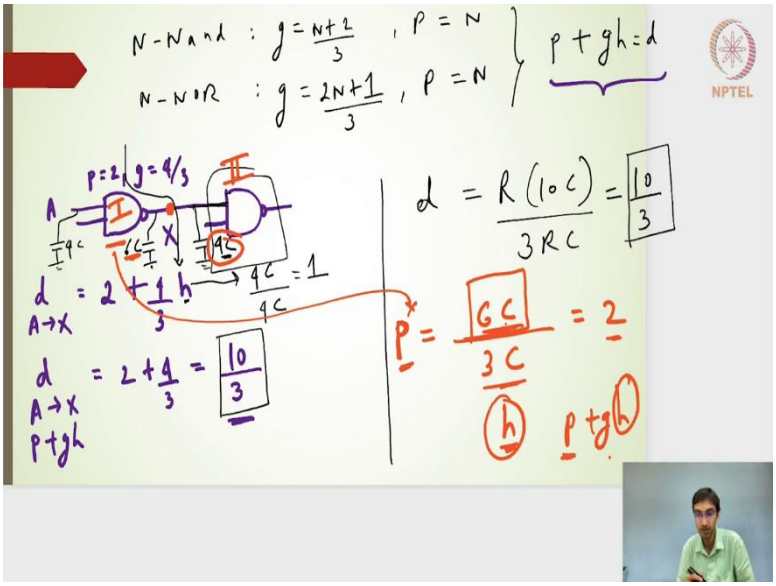
Design and Analysis of VLSI Subsystems
Dr, Madhav Rao
Department of Electronics and Communication Engineering
Indian Institute of Informational Technology, Bangalore

Lecture - 31
Logical effort and Parasitic delay for different gates

(Refer Slide Time: 00:17)



What we had done was for an N input NAND gate, we had said the logical effort will be $\frac{N+2}{3}$ and then the parasitic is N. Similarly, for the N input NOR gates we had said the logical effort is $\frac{2N+1}{3}$ and the parasitic is N. Let us try to validate this result, validate in the sense the linear delay model what we had used, I mean why we are using is to estimate the delay $p + gh = d$, normalized delay.

The absolute delay will be nothing but $(p + gh)3RC$. Let us take a quick example here, if I want to take a very simple example of a 2 input NAND gate, I will have the p value as 2 and the g value for this particular NAND gate is 4/3. Now, this particular NAND gate is going to another NAND gate for a 2 input NAND gates again and this output gets connected to one of the inputs.

What we are interested in this particular A point to this particular X point, the delay of this particular first stage, we are not bothered about the other stage at this particular point of time.

What should be the propagation delay from A to X and I have written a smaller d here, that means that I am interested in the normalized delay.

Using my $p + gh$ model, the linear delay model this particular $p + gh$ model. I will use for this particular delay, for this particular gate of 2 input NAND gates I know that the p value its parasitic delay is nothing but 2 because it is a 2 input NAND gate, g for this particular logical effort of this particular first stage 2 input NAND gate is 4/3 and then h again it is a normalized value.

What should be h here? h is nothing but whatever is the load capacitance coming from the next stage, the load capacitance coming from this particular stage is nothing but it is a 2 input NAND gate, it will give me $2C + 2C = 4C$ of its own parasitic.

If I draw 2 and 2, that will be 2 from the NMOS side and 2 from the PMOS side, it will be 6C. Now just to separate it out 6C and then 4C, 4C is coming from this particular input of the next stage. This h value the electrical effort of the fan out is nothing but whatever is loaded from the second stage, whatever is the load capacitance, which does not include its own parasitic.

6C is not getting involved here in estimating the value of h or the electrical effort because the electrical effort or the fan out is basically considers whatever is loaded. It is not its own stage parasitic capacitance, it is the external load capacitance, that is coming from the next stage the input capacitance divided by its own input capacitance. The own input capacitance is nothing but 4C because it is an AND gate only. I will have 4C/4C, which is nothing but 1.

Finally, what I have is the normalized delay from A to X, using my linear delay model which is nothing but,

$$d_{A\rightarrow X} = p + gh = 2 + \frac{4}{3} = \frac{10}{3}$$

This is my normalized delay, hope this is clear. Let us try to validate this using my own capacitance model, I have written the capacitance model here of 6C and then 4C. The overall delay although I mean I need to find out the normalized delay.

I am going to write the overall delay assuming that the switching resistance here for the charging and for the discharging whatever it is R linear delay effort.

$$d = \frac{R(10C)}{3RC} = \frac{10}{3}$$

This delay is nothing but whatever is the capacitance in the output node which includes the parasitic of 6C and then the 4C capacitance coming from the next stage, which will be 10CR, the switching resistance either falling or rising and then divided by 3RC coming from the inverter, it will be 10/3. This particular value of 10/3 matches with that of this 10/3. Our p + gh model or whatever we have generalize the p and the g factor or the g values or the expressions, it eventually should give us or should lead us to the same delay.

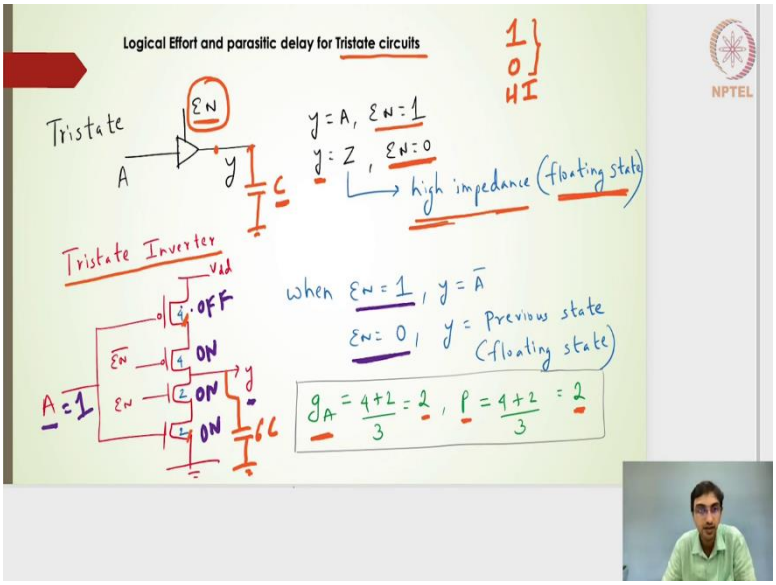
One more important aspect to notice here is because I have cascaded the stages, while cascading the second stage I will use a different color. This particular second stage, I am going to write it as the second stage and this is my first stage. The second stage is going to load some capacitance at the X node, at the output of the first 2. While I am calculating the parasitic normalized delay the P value is by the definition of the parasitic capacitance seen at the node X.

$$P = \frac{6C}{3C} = 2$$

Now this 6C is actually coming or drawn from its own parasitic and we have not included this 4C here. We have not included the input of the second stage into our calculation of our expression for the parasitic delay, that is important. The parasitic delay for the first stage, p + gh for the first stage, the p value, how do we calculate this? This is nothing but its own parasitic capacitance 6C/3C inverters, its own parasitic capacitance giving the same output current for both the gates.

This particular calculation or estimation does not include the next stage input capacitance, that is anyways taken care by the factor of h here. When we do p + gh, this h value or this particular h value is going to consider the next stage input capacitance. This particular parasitic is its own stage the first stage or the second stage whenever we are calculating the delay of the next stage. We will calculate the p value for the next stage, its own stage parasites we will estimate, that will be 6C/3C = 2 for in this particular case, hope this is clear to some extent.

(Refer Slide Time: 08:49)



Moving forward, what we will see is a slightly different circuits, although developed from the CMOS families, but it is called as a tristate circuit. What we will do is we will establish the logical effort and then the parasitic delay for a very primitive tristate circuits and then we will go on to the little bit more accomplished circuits.

What is a tristate circuit? Tristate is defined it consist of 3 states, one is the high state or a low state and then the last one is called as an high impedance state. What it also means is nothing but a floating state, we will come to it. This is the schematic representation of the tristate circuits, given the A input the y should be nothing but A if the enable is enabled or activated or if the $en = 1$ we will have $y = A$, if $A = 0$ we will have $y = 0$, if $A = 1$, I will have $y = 1$.

That is how this 1 and 0 are covered. Now, what is this high impedance state? If $theen = 0$, if it is not activated or the enable pin is disabled (not enabled) at all, the y input is called as the Z state, which is nothing but the high impedance state. It is also called as a floating state because, if I consider the capacitance here at the y output, it will be either be charged or discharged. It will either be charged to V_{dd} or it will be discharged to 0, it will either be holding a value of logic 1 or logic 0 depending on the previous state.

Previous state means whenever the enable was active, whenever the $en = 1$, whatever was the input whatever the output is likely to hold it, that is why it is called as the floating state and the high impedance is coming because you do not have a connection at the enabled side, it is

basically an open circuit the y is not connected to the ground or to the V_{dd} and that is why we say that it is having a very high impedance, the floating state is its appropriate definition.

One can easily visualize if it is a floating state, that means that it should carry the previous state when the $en = 1$ what should be the output y value. In a CMOS circuit representation, the most primitive circuit for a tristate family will be nothing but tristate inverter. The tristate inverter means nothing but I will have the circuits connected like this where the output $y = \overline{A}$.

Whenever the $en = 1$, that means in this particular transistor this will be on, $\overline{en}$ will make this PMOS transistor on, when $en = 1$ and this output is going to either discharge or charge to V_{dd} or discharge to 0 based on the input A, if input $A = 1$ this will be on, input $A = 1$ then this will be on and this will be off.

I will have my output discharge to ground. If $input = 0$ this will be off, this transistor will be off and then the PMOS will be on and then I will have my output charge to V_{dd}. When $en = 0$ here, that means both these transistors will be off, y will be disconnected from the transistors which are connected to the A input.

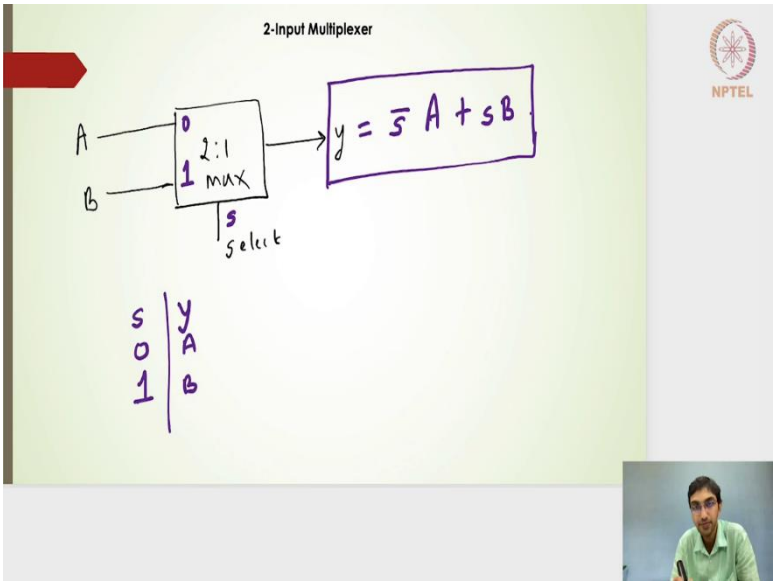
When $en = 0$ this will be off, $en = 0$, $\overline{en} = 1$, that means the PMOS transistor will be off, y will be disconnected from the V_{dd} or the ground rail, it will hold at the previous state. Now, what should be the logical effort of the input A? Input A is connected to these 2 transistors and again by the same definition, if I want the same falling resistance or the rising resistance I will have this sizes accordingly made selected, the size of 2 and 2 because this both of them are in series.

I will have a size of 2 and2 which will give me a falling resistance of R, and the size of 4 and 4 here which will give me a rising resistance of R. Here a 2 and 2 will give me a falling resistance of R, my logical effort for the input A is nothing but this particular size 4 and then this particular size 2/3, that will be 6/3 which will be nothing but 2.

Normalized parasitic delay g_A and p value will be nothing but whatever is the capacitance is here. The capacitance here is nothing but coming from 4 and 2 is 6C capacitance. That is why $\frac{4+2}{3}$. The logical effort and then the normalized parasitic values, we will use the same definitions. For the normalized parasitic it is nothing but the output capacitance, its independent stage standalone gate whatever is its output capacitance divided by the benchmark inverters output capacitance giving the same current.

Both the gate should give me the same output current. That is why we have selected those sizes appropriately. The logical effort again based on the sizes we have to select the sizes, such that it will give me the same output current and the sum of those sizes on the PMOS and NMOS side which divided by the 2:1 inverter's input size will give me the logical effort, hope that is clear.

(Refer Slide Time: 14:42)



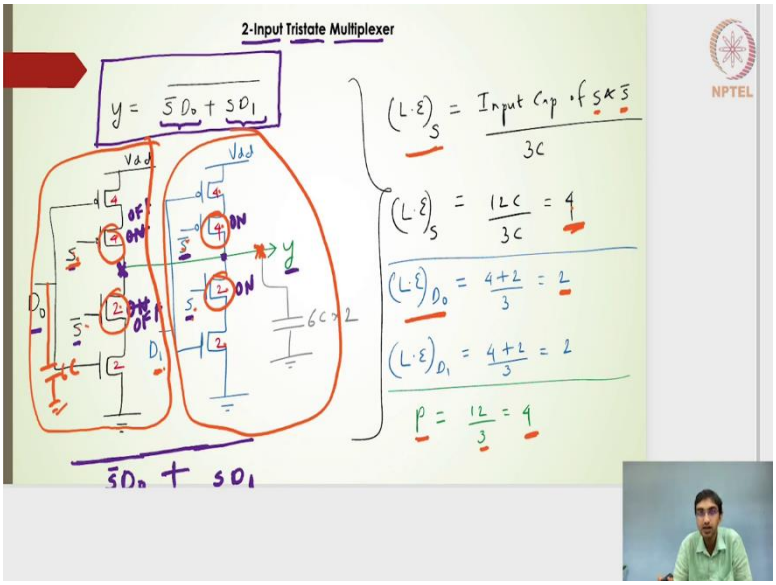
Moving on what we really want to see is tristate is a kind of very very useful for designing the multiplexer circuits and one of the reason is, tristate is very simple to implement. You are going to have a mirror topology, whatever is there implemented on the NMOS side, the same thing is gets implemented on the PMOS side. Whatever is implemented on the pull down side gets implemented on the pull up side, that is one advantage.

The second advantage it simplifies the circuit although the output will always be a complemented output of the input. In both the sense it could be also be the other advantage is, it can be easily be constructed from the CMOS circuits. In that sense the tristate based circuits are kind of very easy to implement and multiplexer is one such application where the tristates are heavily used. But before going into the tristate multiplexer let us try to understand the multiplexer output logic expression.

What I meant was if I want to define the multiplexer. What it means is I have an input, the basic multiplexer will be 2:1 mux. I have A and B as an input and y should be my output and I should have a select line here. I can also say this is the select pin, s pin. Whenever $s = 0$, we

will get y = A. It is nothing but the input pin number 0 and 1. Whenever the s = 1 we will get the output y = B, this is what the basic definition of the multiplexer is, y as an expression I can say that it is nothing but when s = 0, I will get A and then when s = 1, I will get B, this is what we want, going from the tristate, hope you know this is something everyone should be aware of.

(Refer Slide Time: 17:02)



A 2 input multiplexer which is made from the tristate, tristate will always give you an complementary output. What we are seeing is a y is nothing but,

$$y = \overline{\bar{S}D_0 + SD_1}$$

It is a basically an inverted 2:1 mux. How will we draw this? We see this as a 2 branch, $\bar{S}$ and D_0 as one segment and S into D_1 as the another segment. What I meant was segment is, this particular part of the circuit is going to give me $\bar{S}D_0$ and this particular part of the circuit is going to give me SD_1, tied together both of them will give me the complement of it. Let us try to understand the design of the circuit. In this particular part the input D_0 is connected to the transistors PMOS and NMOS transistors which are very close to the rails, Vdd rails and the ground rails.

The select line $\bar{S}$ is connected to this NMOS and S is connected to the complement of $\bar{S}$ will be nothing but S is connected to the PMOS. Whenever s = 0, this will be on, s = 0 PMOS on this side will be on and based on D_0 I will get the output here and on the other segment we have

the D_1 connected to the PMOS and NMOS transistors which are closer to the ground and V_{dd} rails and I have S connected here and $\bar{S}$ connected here.

What it means is if $s = 1$ here this will be on and then $\bar{S}$ will make it 0 and then this will be on. When $s = 1$ this branch will be off. This on will go off when $s = 1$, PMOS1 will be off and $\bar{S}$ will be 0 and that will be off and then this will be on and then this will be on. I will get the output y based on the D_1 input.

When $s = 0$, I will get y is nothing but based on the D_0 signal. It is basically y will be $\overline{D_0}$ or y will be $\overline{D_1}$, that is what the output is. If I connect this both the branch outputs, I will get $y = 1$ of the outputs. Now coming to the sizes. What we are seeing is one of this particular branch will be operative or will be connected to the output. I need to size it accordingly such that I will get y connected to the V_{dd} or y connected to the ground.

If I take this particular segment, I will have the size of these 2 transistors as 2 and 2, the reason is, in the worst case condition this will be off. This will not have any discharging path and I will have the discharging path connected only from y to ground. I will have a size of 2 and2 giving the equal falling resistances R of the 2:1 inverter.

Similarly, for this particular 2 sizes I will have it 4 and 4. So, that one of this path will be on and I will get the rising resistance as R and if I have the sizes as 4 and 4 which will give me a rising resistance of R and then 2 and 2 which will give me a falling resistance of R, that means I will get the equal output current same as that of the 2:1 inverter.

Similarly, if we have designed it for this particular branch we should be able to design it for this branch also very very similar. I will have 4 and 4 and 2 and 2, so that in the worst case condition this will be off and then this will be on and similarly when this is on this particular path will be off, I will have a size of 2 and 2 and 4 and 4.

If I have this selected size of 4 and 4 and 2 and 2 on both the branches, at the output side if I want to find out the parasitic capacitance, I need what is the overall parasitic capacitance seen at the output node y. Turns out to be nothing but this 4, this 4, this 4, this 2 and then this 2 will contribute to overall I will get 6C x 2 which is nothing but 12C.

The normalized parasitic delay will be nothing but,

$$p = \frac{12}{3} = 4$$

This 3C is now our 2:1 inverter parasitic capacitance. The logical effort, I mean there are D_0 is one input, D_1 is another input, S is another select signal. If I want to find out the logical effort of D_0, it is nothing but its capacitance seen here which is coming from the PMOS, another one coming from the NMOS,

$$(LE)_{D_0} = \frac{4+2}{3} = 2$$

Logical effort of D_1 will be,

$$(LE)_{D_1} = \frac{4+2}{3} = 2$$

For the select line,

$$(LE)_s = \frac{\text{Input Cap of S and } \bar{S}}{3C}$$

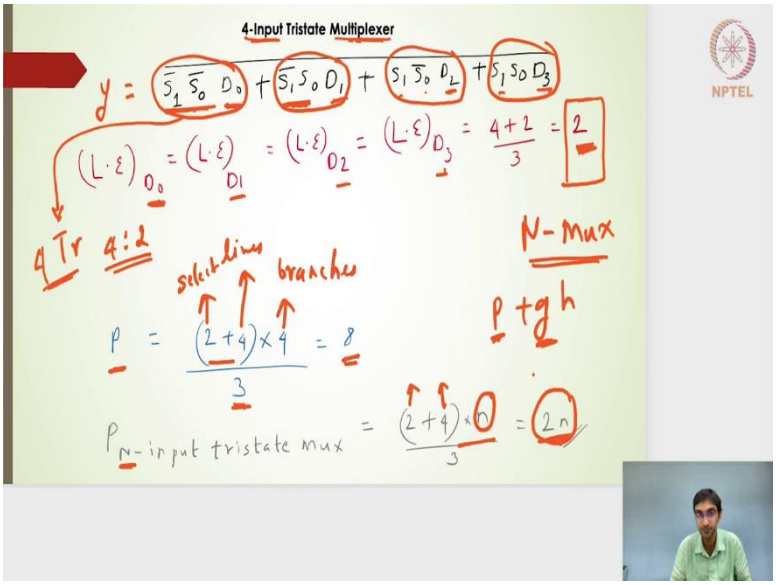
In this case for the select line S and $\bar{S}$,

$$(LE)_s = \frac{12C}{3C} = 4$$

What we are saying for the select line is the logical effort, for the select line is we are considering the input capacitance of S, not only S but also for the $\bar{S}$ and the reason is very very simple is because this $\bar{S}$ will be generated from the S signal somewhere in the circuit. Somewhere before the circuit we will have an S signal directly connected to this particular circuit, this particular module $\bar{S}$ will be there somewhere generated from an inverter and if we do not know the size of that particular inverter it is better to take its own sizes here in this module, it is a very conservative approach and then calculate and estimate this particular g value for estimating the delay. In that sense we are taking all the input capacitance of not only the S as well as its own complemented input, I will have S and $\bar{S}$. It will be 12C/3C which will be nothing but 4.

Hope this is clear, the logical efforts and the normalized parasitic delay for the 2 input tristate multiplexer.

(Refer Slide Time: 24:23)



Continuing further what should be the 4 input tristate multiplexer? If I have 4 input, we will calculate, we will have an expression like this,

$$y = \overline{\bar{S}_1\bar{S}_0D_0 + \bar{S}_1S_0D_1 + S_1\bar{S}_0D_2 + S_1S_0D_3}$$

S_1, S_0, I will have a D_0 signal, when for 0 1 case it will be D_1, for 1 0 case it is D_2 input and then 1 1 case it is D_3 input.

There are 4 inputs D_0, D_1, D_2, D_3 which gets at the output based on the 2 input select lines. S1 and S0 are the 2 input select lines, this is the expression. I will have basically 4 branches, one branch is this, another branch is this, another branch is this and another branch is this and this 2 input is going to one NMOS transistor and one PMOS transistor. This D_0 is going to one NMOS and one PMOS.

In each branch I will have still 4 transistors, again these 2 input combined will go to one NMOS and one PMOS transistors, of course its complement will go to the PMOS transistors, again complement will go to either PMOS or NMOS and D_1 will go to the transistors PMOS and NMOS transistor which are closer to the rail.

I am still imagining or visualizing this as a individual branch as a 4 transistor circuit, 2 transistor on the pull down side and 2 transistors on the pull up side and we will have the same sizes

because it is still the 4 transistors, either one of these branches will be on. I will have the PMOS to NMOS transistor ratios will still be 4:2.

The logical effort of all the input signals D_0, D_1, D_2, D_3 and I am not taken the select line logical effort, it is only the input signals D_0, D_1, D_2, D_3 will be nothing but,

$$(LE)_{D_0} = (LE)_{D_1} = (LE)_{D_2} = (LE)_{D_3} = \frac{4+2}{3} = 2$$

The normalized parasitic delay, because I have 4 branches now the output node I will have,

$$p = \frac{(2+4)x4}{3} = 8$$

This is the number of branches and then this 2 + 4 is coming from the transistor which are closest to the output rail. It will be the input fed to this particular transistor will be the select lines and its complement. Those are I will call it as the transistors driven by the select line signal, $\frac{(2+4)x4}{3}$, 3 is coming from the benchmark inverter 2:1, benchmark inverters output parasitic capacitance.

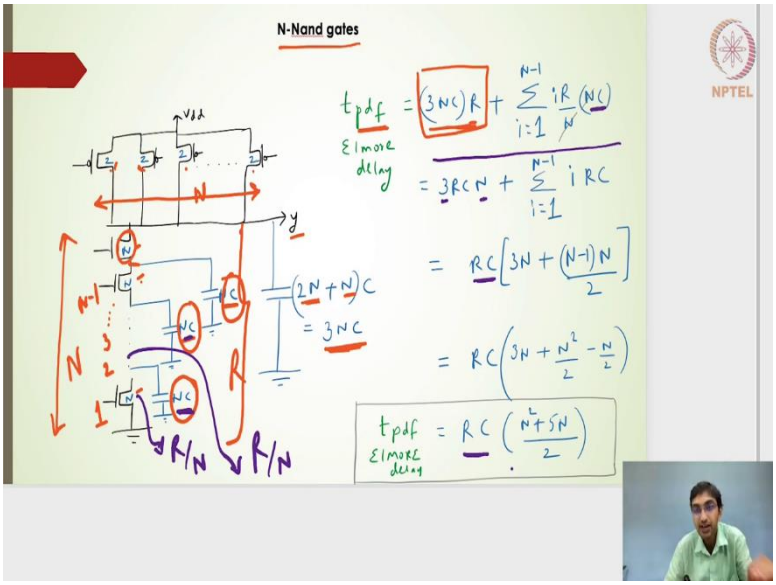
If I look into the 2 input tristate multiplexer. If I go back the parasitic was 4, for a 2 input tristate multiplexer and if I come back to this current slide for a 4 input it is actually 8. For an N input tristate multiplexer it is nothing but,

$$P_{N-\text{input tristate mux}} = \frac{(2+4)xn}{3} = 2n$$

This 2 and 4 is coming from the select line, the transistor and this n is nothing n number of branches or the n number of inputs, which will give me 2n parasitic. Remember that the logical effort still remains 2 here for the inputs in the 2 input tristate multiplexer it was still 2. The logical effort does not change for an N input multiplexer if you are doing designing via the tristate, that is a clear cut advantage. I am having a delay $p + gh$, this g value for an input does not change. Whereas, the p of course, it will change linearly with respect to the number of inputs, but this g value does not change.

If I design it in another way, we will have a change in p, we will also have a change in g. In this particular case when we have it branch wise based on the number of inputs the g value does not change, hope this is clear.

(Refer Slide Time: 29:01)



Just to complete the overall the logical efforts and then the parasitic delay. Now basically the linear delay model what I have done here in the next 2 slides including this one will be trying to showcase the Elmore delay model based delay estimation and compare with that of the linear delay model with respect to the number of inputs.

I have taken a simple example of an NAND gates, which is an N input NAND gates. For an N input NAND gates, I will have N such PMOS transistors and N such NMOS transistors and I will select the size of N here on the NMOS side. Because I need an equivalent falling resistance to be R giving the same output current of I as that of the 2:1 inverter.

On the PMOS side the sizes should be nothing but 2, but it will have N such transistors. At the output node y, I will have each of this PMOS transistors contributing a capacitance of 2N and this NMOS transistor N closest one to the output rail, the transistor which is closest to the output node will contribute to NC capacitance, the overall I am seeing a 3NC capacitance.

Using an Elmore delay method signifies that I need to account for all the inter node diffusion capacitance NC, NC and NC all the intermediate diffusion node capacitance, I need to consider that while estimating the delay. What it means is I have taken the propagation delay for the following, that means for the N input NAND gates all the N inputs here on the pull down side should be 1 so that the output will discharge to 0.

$$t_{\substack{pdf \\ elmore \\ delay}} = (3NC)R + \sum_{i=1}^{N-1} \frac{iR}{N}(NC)$$

I have taken 3NCR, R is nothing but the switching resistance of all these N transistors which are in series will give me a switching resistance of R equivalent falling resistance as R. It is 3NCR, and then I need to accommodate for this particular capacitance plus N plus this particular capacitance and so on.

This one 3NC is already accommodated here and I am left with now N-1 transistors starting from this one transistor, then the second transistor, then the third up to this N - 1 the N transistors its own parasitic is anyways accommodated in this 3NC. What I have done is, I have taken a summation signal and I is going from 1 to N - 1 transistors and its parasitic diffusion intermediate diffusion node capacitance is NC and its equivalent resistance in this case I am taking it as iR/N. Each of these transistors is showing a resistance of R/N, this second transistor should also should give us an R/N.

But as we are proceeding from down to up, closer to this particular transistor of NC, it sees a resistance path of R/N, the next one sees a resistance path of 2R/N, the third NC capacitance sees a resistance part of 3R/N that is why I have made it a variable of iR/N. If I have written this particular generic expression for the Elmore delay then further calculating,

$$t_{\substack{pdf \\ elmore \\ delay}} = (3NC)R + \sum_{i=1}^{N-1} iRC$$

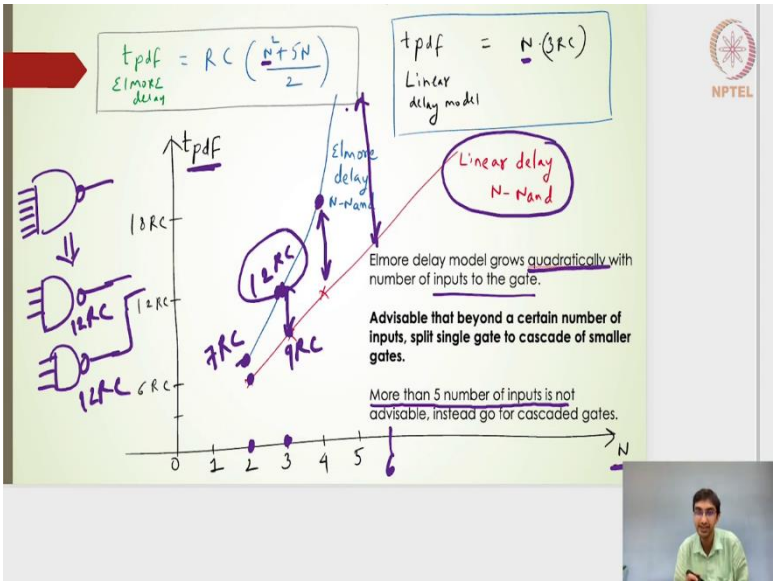
$$= RC\left[3N + \frac{(N-1)N}{2}\right]$$

$$= RC\left[3N + \frac{N^2}{2} - \frac{N}{2}\right]$$

$$t_{\substack{pdf \\ elmore \\ delay}} = RC\left[\frac{N^2 + 5N}{2}\right]$$

This is my Elmore delay propagation delay falling using the Elmore delay method.

(Refer Slide Time: 33:28)



If I use that particular expression to draw a profile of the propagation delay falling with respect to the number of inputs. If I keep N is equal to you know if let us start from a 2 input NAND gate, that is the most primitive NAND gates we have. For a 2 input NAND gates I will get $\frac{N^2+5N}{2} = \frac{2^2+5x2}{2} = \frac{14}{2}$.This will be nothing but 7RC absolute delay, whereas a linear delay model is going to give me the output node capacitance or rather the propagation delay falling will be nothing but whatever is the parasitic capacitance multiplied by the 3RC. The parasitic capacitance is nothing but we know that it will be N the output node capacitance. The parasitic capacitance seen at the output node we ignore the inter node diffusion capacitance completely in the linear delay model, it will be NRC.

For a 2 input NAND gate, it will start from 2x3RC which is nothing but 6, for a Elmore delay method it gives me slightly one RC above, there is a deviation. If I go for it N = 3 input, I know that this value will be 9RC on the linear delay model, on the Elmore delay method it turns out to be 12RC.

There is difference you know which was 1RC here in the 2 input NAND gate, now it grows to 3RC in the 3 input NAND gate. In the 4 input NAND gate the difference is quite more, in the 5 input NAND gate the difference is even more and so on. The difference between the Elmore delay method which is more appropriate than the linear delay model keeps on growing. The linear delay model what it says it is a of course, it is a non-linear delay model it grows quadratically with the number of inputs.

The N^2 here, as the N increases the number of inputs coming to the number of inputs that defines the N input NAND gate if it keeps on increasing. The Elmore delay method shows that the delay increases quadratically to the number of inputs to the gate. The advisable is beyond a certain number of inputs, it is not advisable to have more number of inputs in a gate.

It is advisable to keep that number of inputs to a limit, what is advisable as given in the textbook Weste and Harris, if possible design the circuit still the number of inputs reaches the 5, beyond 5 if I use a 6 input NAND gate the difference is very very huge and if I am doing a rough calculation by a linear delay an input NAND gate the difference is huge that is 1.

The second thing is, if I consider the more appropriate model which is an Elmore delay method, if I use a 6 input NAND gate it will be somewhere here, even beyond that the delay is actually very very high. If I can resolve that the 6 input NAND gate, let us say a 4 input NAND gate or a 3 input NAND gates 2 such 3 input NAND gates if I cascade it.

If I have instead of 6 inputs, if I can actually do it using 3 - 2 such NAND gates and that will be much more preferred, because my delay of 12RC as compared to quadratically increased 6, it will be very very high. Instead of that having a lower 12RC delay from 1 and 12RC delay simultaneously and then getting it connected to the next stage.

It is always advisable to keep the number of inputs connected to the NAND gates or the NOR gates limited to 5. One thing is when I have more than the 5 number of inputs the linear delay model estimation will be completely be off and the second thing is the Elmore delay method which is more appropriate, but still as a delay increases quadratically you know, as a delay increases quadratically with the number of inputs it is you know even if you use an Elmore delay method it is going to be very very high.

Why not reduce the number of inputs and then redesign the circuits, we will need the same output, but by using multiple such NAND gates or multiple such NOR gates, but having less number of inputs to those gates.