Advanced VLSI Design
Prof. A. N. Chandorkar
Department of Electrical Engineering
Indian Institute of Technology- Bombay

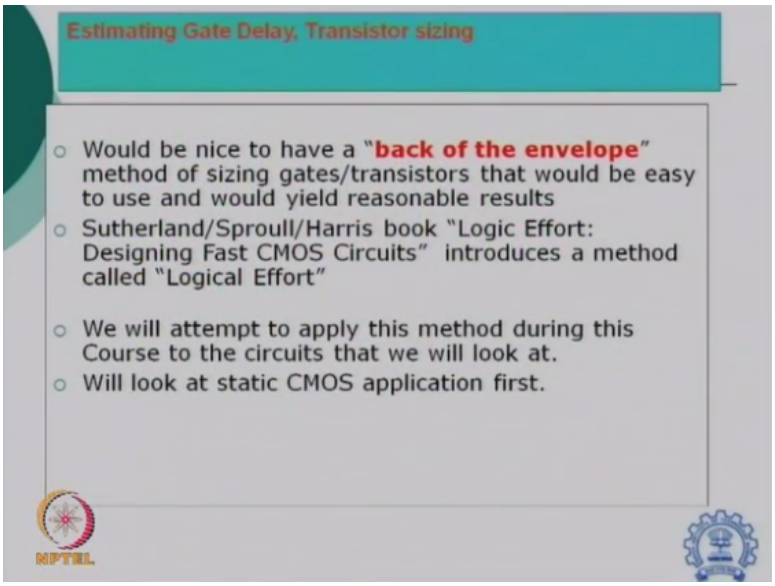
Lecture - 04
Logical Effort-A Way of Designing Fast CMOS Circuits (Contd.)

Last time, we were trying to see what is the importance of logical effort and we actually discussed something about how to calculate the propagation delay of CMOS inverter and based on that I suggested last time that we will come out with a newer and simpler technique to design high-speed circuits where average production delay from input to the output of a circuit is of relevance which decides the speed of the data flow.

Therefore, last time we discussed about actual transfer evaluations and we figured it out that the width of the transistor decides the speed essentially because it decides the capacitance and it also decides the current charging or discharging those capacitances. So, now today we shall start with the logical effort method suggested by Sutherland Sproull and Harris in their book which is essentially published by Morgan and Kaufmann and is available in the market.

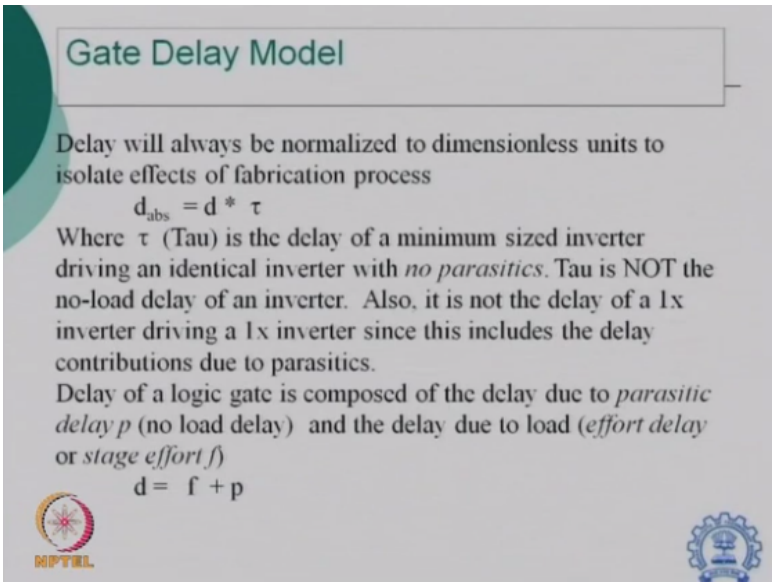
Please do have a look at it if you can. Now here we will attempt to apply this method during the course of this circuit that we look at and we will also look at static CMOS application first.

(Refer Slide Time: 01:31)



Now, let us start with what we said.

(Refer Slide Time: 01:35)



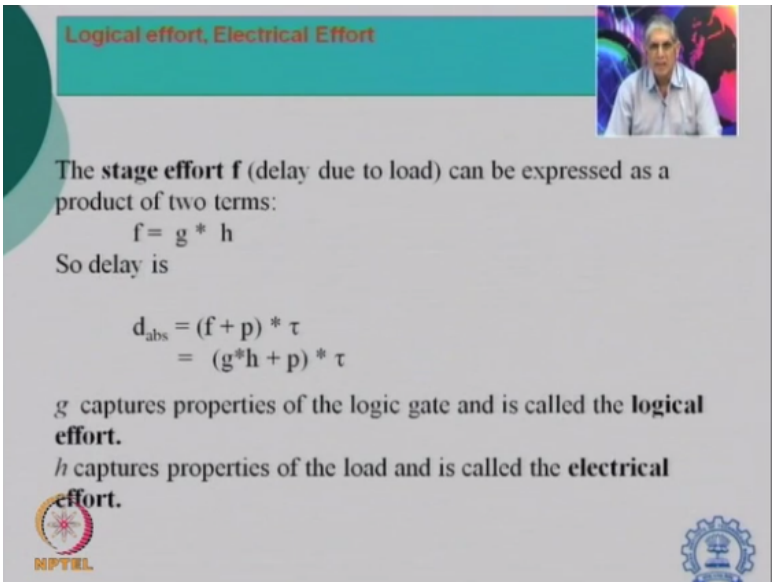
We start with a model which we call gait delay model. Now, what does it says, delay will always be normalized to a dimensionless units to isolate effects of fabrication process. Because you know the Tau which is essentially the minimum size invertent which is driving another inverter with no parasitic is essentially decided by some kind of technology parameter. Therefore, we will actually not use Tau as any parameter in our evaluation.

Actually, we will work only on d which is d absolute divided by Tau. So, it is normalized to Tau daily is d and that is the delay we are going to work with. Please remember Tau is not no load delay of an inverter. It is essentially an inverter driving another inverter which is identical with no parasitic. Please remember these are some terminologies which Sproull Sutherland others have decided.

Also, it is not delay of a 1X inverter driving 1X inverter since that includes the delay contribution due to parasitic. So, please be clear about the definition of d, definition of Tau, and therefore we will be able to evaluate the actual delay of a circuit which is normalized delay d. Now, delay of a logic gate is composed of delay due to two parts; one is due to the parasitic delay p which we say no load delay and delay due to the load which is effort, we will call that as an effort delay or a stage effort f.

So, one is because of the delay due to the load which is we say f and then delay due to the parasitic which we say p, so the net delay of course again normalized net delay d is f + p.

(Refer Slide Time: 03:18)

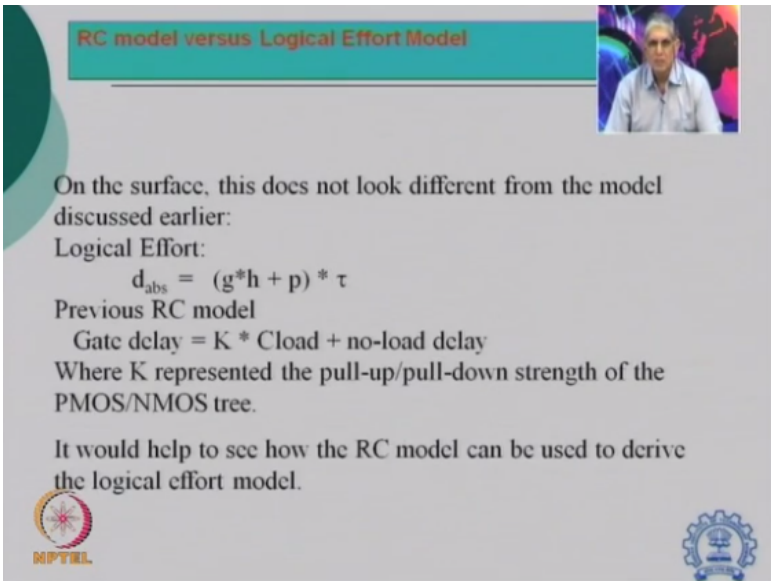


Now, this stage effort f which is delayed due to the load can be expressed as product of two terms. F is for example given here is f = g x h, where these two terms what is g and what is h is of relevance and we will see quickly what they meant. So, if we substitute in our delay model this term f, so we get d absolute is f + p x Tau or equal g into h + p x Tau. So, if we want only d then it is g into h + p and that is the actual delay we are interested to evaluate.

Now what are these term g and h. So, it is defined as g captures the properties of any logic gate and is called the logical effort which is most important part in all this analysis which we follow now. I repeat it captures the property of a logic gate and is called logical effort. Whereas, that means it is property of the gate itself. Whereas the other term h actually captures the properties of the load and therefore called electrical effort.

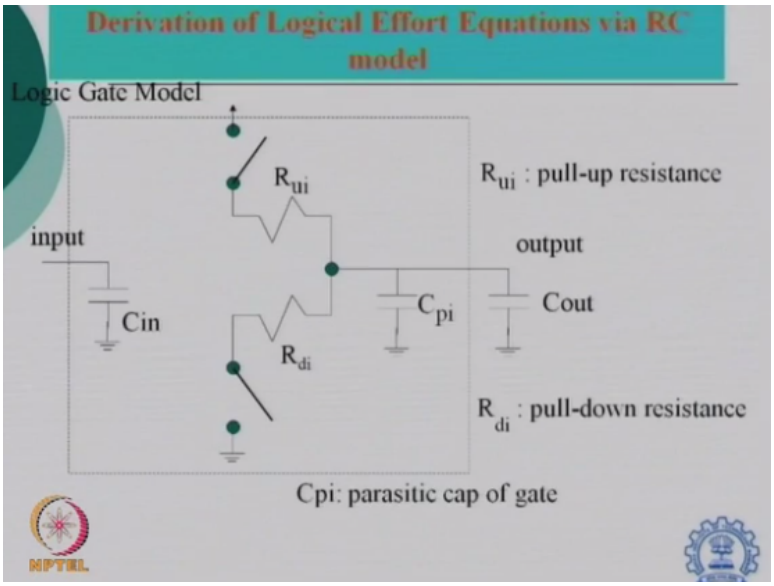
So, it is external to the gate is the load and therefore we call any value of load is actually taken care through term h. Any property of the gate through which current and capacitance are evaluated is calculated through term g.

(Refer Slide Time: 04:44)



On the surface of this does not look very different from the model already discussed by us. So, we say logical effort is gh + p x Tau, whereas our previous RC models which are very popular as simple as that is net gate delay is some constant K which represents the ratio of pull down to pull up transistors. So, k times the net capacitance is load capacitance plus no-load delay. Now, it would help to see how a typical RC model can be used to derive the logical effort model and that is what now I am going to do.

(Refer Slide Time: 05:22)



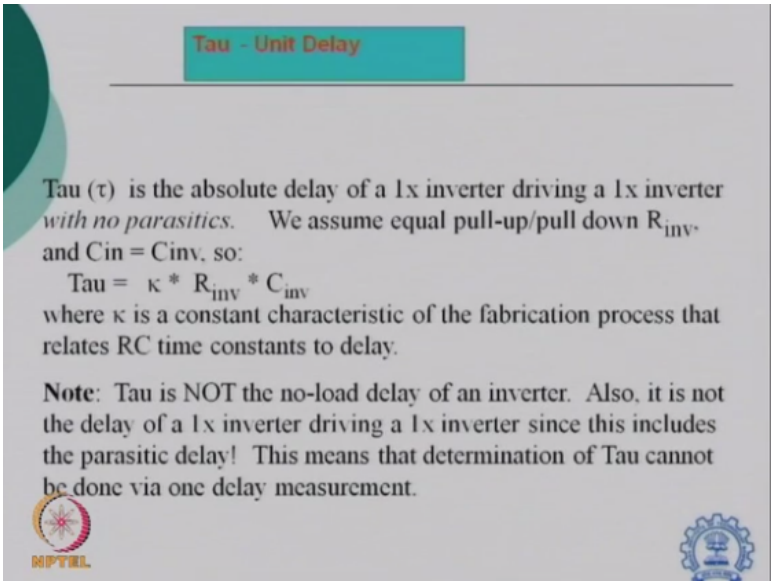
It is a typical logical effort equation with why RC model is show here. You can see the pull-up transistor okay has an equivalent resistant which it has, we can see here Rui and Rdi are pull-up and pull-down resistances of the transistors. Pull-up transistor in given P channel kind and pull-

down transistor in channel kind and essentially we say output of this inverter which they call input to the capacitance which will actually charge the output capacitance Cout which is essentially the net load will be parasitic capacitance plus output loads.

So, net ACL which is CPI + Cout. Now, we are shown two switches, both towards ground side. What it means essentially when the P channel is on in the case of switching of a CMOS inverter, then a P channel is on means the upper switch is connected and the lower switch is open. When the pull down starts, we say the upper switch is open and lower switch is connected, so that the capacitance can discharge through RDI to the ground.

So, this is typically RC model which we will be using in deriving what we call the values of g and h.

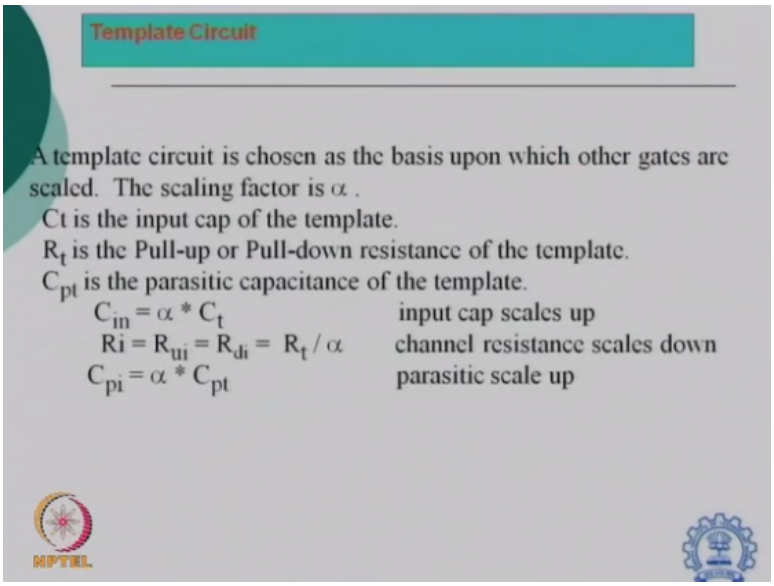
(Refer Slide Time: 06:34)



Now, I repeat again, Tau is absolute delay of a 1X inverter driving a 1X inverter with no parasitic. We assume equal pull-down pull-up ratio which we call Rinv and Cin is essentially the input capacitance of an inverter, we say Cin, so that the Tau can be written as Rin x Cin x K which is actually a constant of fabrication process. So, essentially it is RC delay and Tau is K times R inverter x C inverter.

Please remember once again I am repeating, Tau is not the no-load delay of an inverter, also it is not the delay of 1X inverter driving 1X inverter because that may include parasitic delay. Here, we essentially means Tau is 1X driving 1X without parasitic.

(Refer Slide Time: 07:24)



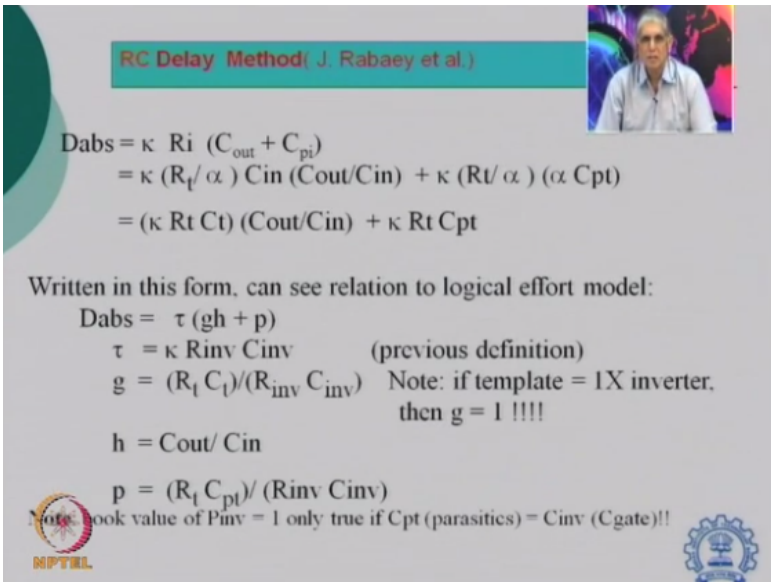
Now we say a template circuit is chosen as the basis upon which other gates are scaled. The scaling factor is alpha. If you see very carefully in the normal CMOS inverter, the P channel device is as twice the size of an N channel device, channel link being same for both device. See, the width of P channel is normally kept double that of N channel. This ratio is appearing because we want symmetric transitions and to have equivalent resistance on both sides.

We know mobility ratio of Mu N to Mu P is 2, so to compensate for that we actually double the size of P. But we can have any other factor, Mu N Mu P may not be two. So, we say that scaling factor we call it as alpha. Now, CT is the input capacitance of the template circuit. RT is the pull-up and pull-down resistance of the template, CPT the parasitic capacitance of the template. Then, we say CIN now is alpha x CT which is our template capacitance.

RI which is now input resistance which is equal to RUI or RDI because either of the P Channel or N Channel will be on. So the resistance RUI = RDI and that is equal to RT template resistance divided alpha which are scaling and CPI which is the parasitic capacitance is alpha x CPT. So,

what it essentially means, when you scale alpha is greater than one, input capacitance scale is up. Channel resistance scale is down, and parasitic capacitance scale is up.

(Refer Slide Time: 09:06)



So, if we see now d absolute from our simple theory, then we say it is equal to K Ri x Cout + CPC RIN you can say. RIN x Cout + CPI are net capacitance, then I replace RIN, Cout, and CPI in the expression from the expression I wrote earlier. So, I say it is K x Rt x alpha which I represent RIN and Cout + CPI rewrite as CIN x Cout/CIN plus K x RT/alpha and alpha x CPT. Doing little simpler math, we say d absolute is KRTCT x Cout/CIN + KRTCPT.

Now written in this form, we can see relation to logical effort model now. We have said absolute delay is Tau into GH + B. Now, Tau there we said K into R inverter into C inverter. G = RtCt/R inverter/C inverter. H = Cout x CIN and P = RtCpt / R inverter C inverter. So, the definition from this is clear to us that the logical effort term G is essentially the ratio of RC time constant of your template circuit divided by your normal inverter circuit.

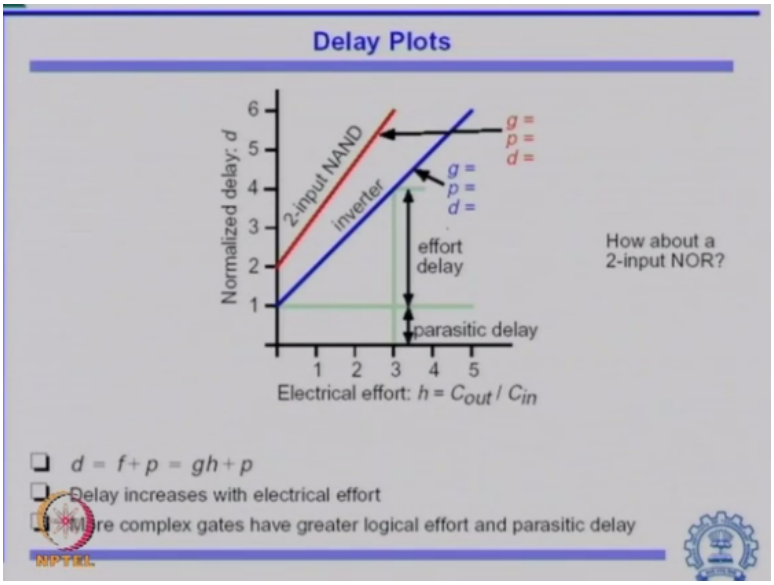
RIN x CR are time constant on normal inverter, R inverter x C inverter. So, one can see clearly from here, if you have a 1X inverter that means RT is RIN and CT is CIN which is equal to one. So, for a normal inverter template and the normal inverters are same, then G is equal to one but if your inverter RTCT is not same as our inverters, inverter G will be higher than one and we like to calculate how much is this logical effort G for different kinds of our circuits.

Please remember here that the H value you are seeing is the ratio of the output capacitance to the input capacitance. What does it mean, it is immaterial that in the chain of blocks of inner circuit, the between capacitances are not relevance in calculating the H. We are only interested in the input capacitance decide the first driver stage and output capacitance decide the output stage of that inverter.

So, obviously how much current I should provide through inverter to charge the Cout if CIN is so much, and one knows very well that if Cout is very large compare to CIN as in the case of buffer stage and then to provide large current for a faster circuit to operate, you ought to have a larger current to be provided by the inverter or buffer and if you larger sizes of the inverters, then obviously the input capacitance being larger.

Because there width will be larger in both P Channel and N Channel. So, the ratio of Cout to CIN essentially is the one which decide the speed at the end of the day. Also we know that P is equal RTCPT divided by inverter and typically we say P inverter value is normally taken as one if parasitic capacitance is equal to normally one inverter capacitance. Here is some interesting graph shown.

(Refer Slide Time: 12:34)



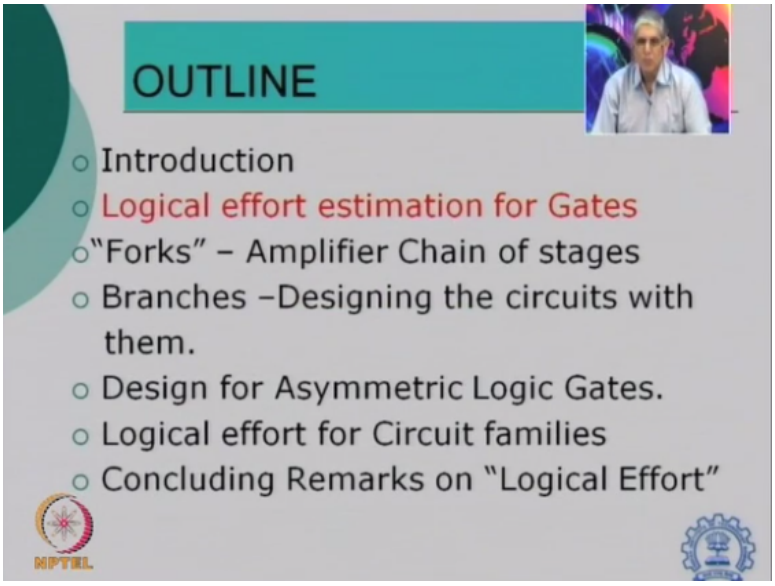
We know D is equal to F + P which is GH + P. So, if I plot electrical effort versus normalized delay, D is not the absolute delay but D absolute x Tau, so it is a normalized delay and you see the blue line which is for inverter which shows that since I said inverter has a parasitic delay of one, so even if there is H equal to 0, there is a normalized delay D equal to one. So, P is equal to one and H is 0.

So that is the intercept I am showing you on the Y-axis and as I increase H since it is a straight line, you could see that the delay will linearly rise with the blue line shown here, and if you see now from the parasitic delay line at 1, above whatever is the additional delay you are getting is essentially because of charging the additional capacitance of the load and that is taken care through what we call electrical effort and that is the delay essentially called effort delay.

More complex grid have a greater logical effort and parasitic delay and that is what we are going to see. So, if you see equivalently another device, though we will prove this point. If you have a 2-input NAND gate, it can be shown that it is parasitic delay is twice, 2-input stand for one each input, so a parasitic delay of normalized delay is 2 and it has a slope which is different from inverter.

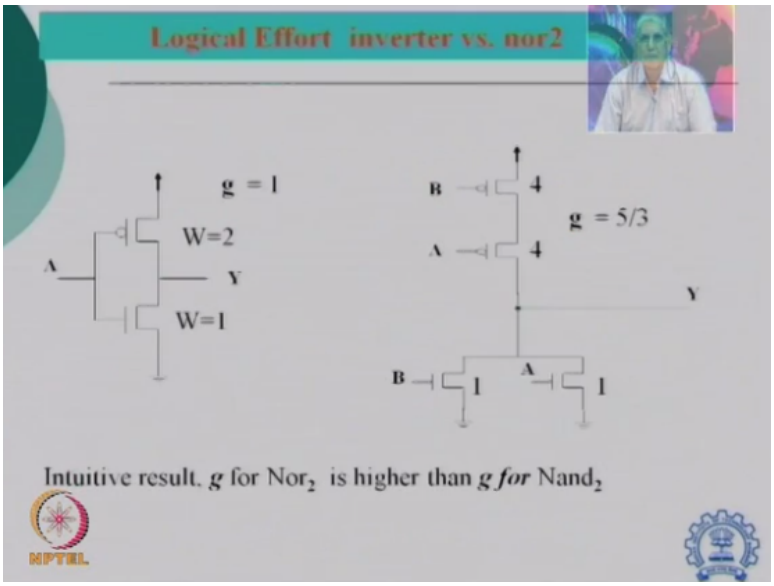
This is most important because in NAND gate we will see in a static case at least, you have two inverter chains, two N channel and P channel transistor in series and their net capacitance will be larger and therefore even at a lower logical electrical effort, we may have a larger delay. We will see into this very soon.

(Refer Slide Time: 14:25)



So, now having given you some introduction what is the logical effort terminology we are going to use, we now look into logical effort estimation for different gates which is very important because at the end of the day if I know my G then if I know my H, I will be able to calculate what is my delay and delay in essentially is related to speed by one upon delay is speed and therefore we are interested to know what is the net electrical effort H which is COUT x CIN.

(Refer Slide Time: 15:01)



So, let us look at the logical effort in inverter versus a NOR gate. When I say I am having a standard gate or template gate which I am going to decide with which I will scale everything, so here is my left here is a standard CMOS inverter where width is 1 and here P channel width is 2,

assuming channel lengths are same which has to be true in technology. So, we do not say W by L, we only say width as twice and width as 1.

And one can see this as I just now said if are we compare this inverter with itself, then the G which is the ratio of RC x 2 will always be same and therefore we say G is equal to 1. However, here is a NOR gate shown to you. You have two P Channel devices and series and you have two N Channel devices in parallel. This stands for a simple static NOR. Okay, now the idea in all derivations is following.

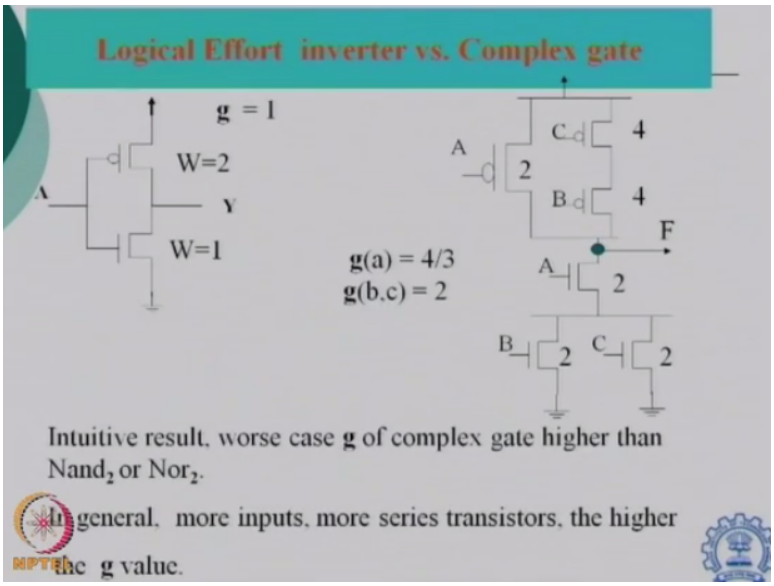
If you have a parallel combination, the minimum size transistor width, width one is the minimum, so you keep each transistor of the minimum this which is then equivalent of this case. Because in the case of NOR, there will be either this branch operating or this branch operating. In either case therefore you need at least one equivalent of that. However, if you look at the P channel device, since there are two transistors in series.

And if I want to make equivalent of W equal to 2, obviously I must make each transistor double that of normal P Channel which is four times, W is 4 here, W is 4 here. Series combination therefore will give equivalent resistance of only equivalent of 2 and therefore now I have a situation when I have two P Channels with width of 4 each in series and two N Channels in parallel with a width of one.

For each input A and B, one can then evaluate capacitance, which are as I said you already, already proportion to their widths. So, we can see from here for A input the capacitance in proportion to four units, for A input capacitance in proportion to one unit, so obviously one can say, the net capacitance is due to input A is proportion to 4 + 1, so we say it 5 and if you see a normal inverter here, the net capacitance will be proportion to 2 + 1 which is 3 while template capacitance proportionality 3.

So, ratio of this to this is therefore 5 x 3. For each input, this is 5 x 3. For B also it will be 5 x 3, so essentially if you say it net logical effort for a 2-input NOR gate is twice that of 5 x 3 or which is 10 x 3.

(Refer Slide Time: 17:59)



Now, we look into other similar simpler box. For example, by logic of same we can see here is a complex gate which has a 3-input ABC. Essentially, what we are doing is this is actual function which you are implementing is A + BC bar in the complex function which we are implementing here, so we see for A.B + C bar is the function implemented. Since it is a B + C 2 are in parallel and for input A which is here in series to that.

Now interesting feature is you can see from here since each template has to have width W and since these two transistors are in parallel, so obviously at least each arm should give you W equal to average equal to 1 which is essentially capacitance on 1 each line should be 1. But one can see from here since these two will be in series or these two will be in series, so if I had make N Channel chain as width equivalent of W1.

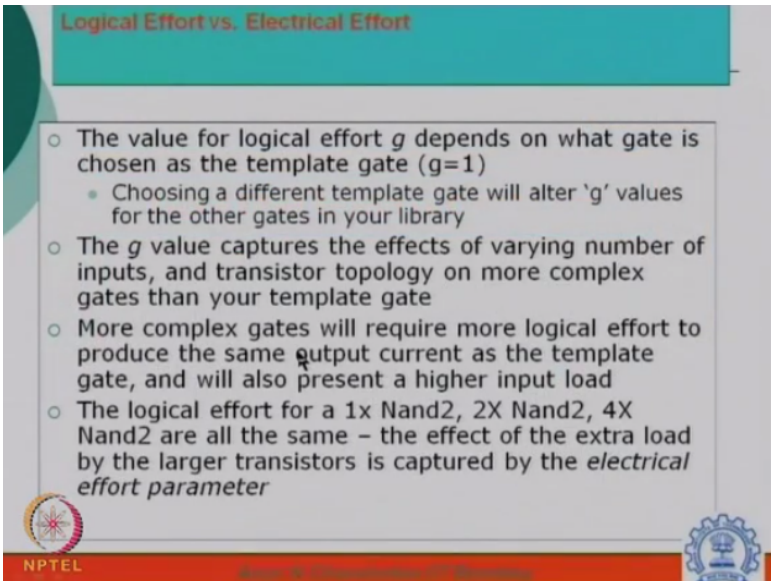
These two at least should do in series combination such that the net W is 1, so 2 and 2 must be kept so that parallel series combination of 2 and 2 will give me W equal to 1. Similarly, 2 and 2 here will give me W equal to 1. So, now I say A will have width of 2, B will have width of 2, and C will have width of 2. So, that this is equivalent of a pull-down chain of W equal to 1. If you look at the P Channel pull-up area.

We know it is opposite of this occurs, B and C will been in series and A will be parallel to that; however, since you need W to firm to power supply to the ground, so obviously from power supply to the output, at least each arm this and this should give me equivalent of W equal to 2 since does not have anyone in series. Each arm, this and this should give me equivalent W equal to 2, since A does not have any one in series.

So we say A must have a width of 2, whereas C and D, B and C should have width of 4 because they are going to be in series, so 4 and 4 in series will give me 2, so path is either this 2 or this 2 to the output. Now, if I see per input as the logical effort if I evaluate, I say A has this 2 and this 2, so it is 4. This is of course 3, so I say, 2+2=4/3 is g logical effort for input A. By similar logic, logical effort for B and C you can see from here now, this is 4 and this is 2, so 6/3=2.

So logical effort due to input B and C, both because they are similar, we say it is 2 and the net logical effort of a such a complex gate will be 2+2+4/3, okay. So indicatively result what we are saying, the worse case g of a complex gate is always higher than the Nand 2 or NOR 2 gate, okay.

(Refer Slide Time: 21:13)



The value for logical effort g depends on what gate is chosen as the template, in our case, we have chosen 2 is to 1 inverter as our template gate and therefore we say cord g=1. However, please remember this is simplicity which has made us to think that inverter is equivalent of that

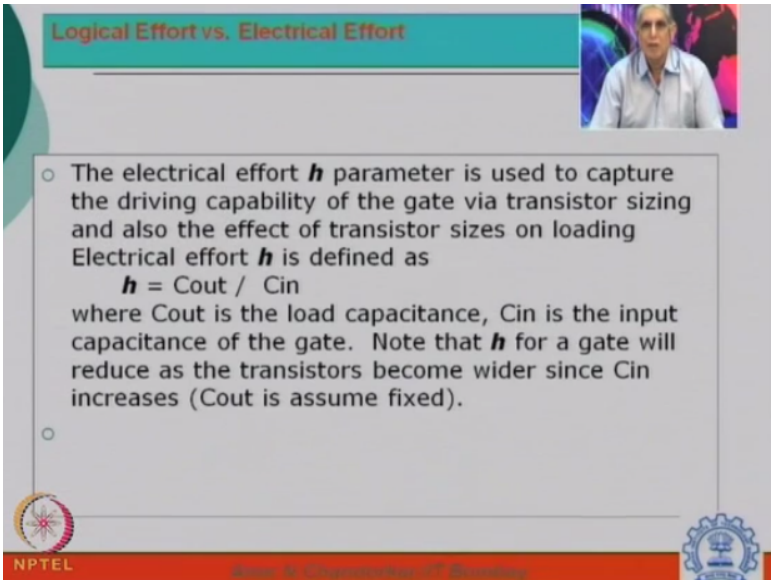
but otherwise you can chose any other template gate and all that will have that will call a some value g for that.

And with reference to that, we can scale other circuits or we find what is the additional effort coming out of that and you will get another library of function with different logical efforts. So what essentially g is doing, in my opinion, the g value captures the effect of varying number of inputs A, B, C and transistor topology which is with a series combination or parallel combination and compared to the template gate.

More complex gate will require more logical effort to produce the same output current as the template gate and will also present a higher input load. The logical effort for 1x Nand2, 2x Nand2, 4x Nand2 are all the same; however, the effect of extra load by the larger transistor is captured by electrical effort parameter but not by g. So Nand will have, whether it is 2x Nand or 4x Nand axis strength, it does not matter.

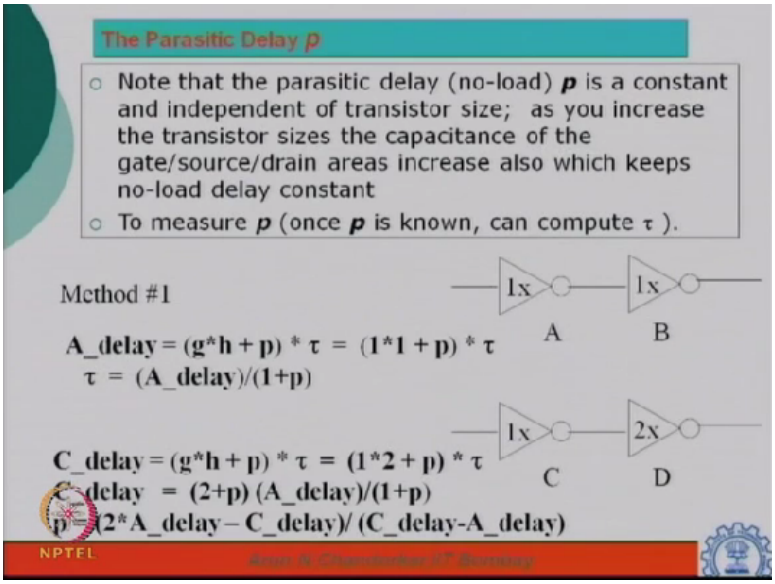
Whereas the increase of size essentially increases the input capacitance and also the output capacitance is and therefore, the ratio of that we will decide through the electrical effort parameter h.

(Refer Slide Time: 22:51)



So I repeat, electrical h parameter is used to capture the driving capability of the gate via transistor sizing and also the effect of transistor sizes on loading; therefore, we say h = Cout/Cin, where c out is the load capacitance and Cin is the input capacitance of the gate. Please note that h for gate will reduce as the transistor becomes wider since C in increases, C out is assumed fixed at the output.

(Refer Slide Time: 23:20)



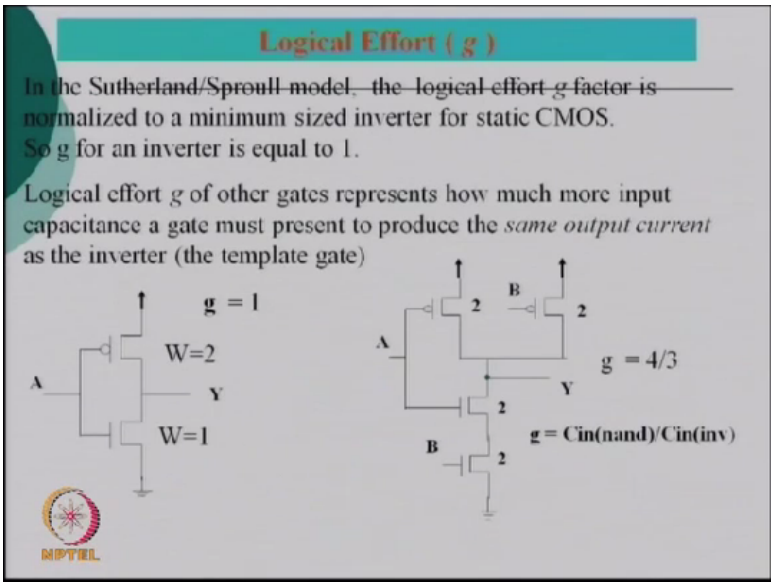
Now there is some definition of parasitic delay p, okay. Note that Parasitic delay p is constant and independent of transistor size, as we increase the transistor sizes, the capacitance of the gate shows drain areas increase, also which keeps no load delay constant. So how do we measure p, so we must find out how to monitor p but once known, we can then calculate Tau very easily. So we have 2 methods to suggest, method 1 says you have a circuit shown here.

1x inverter driving another, A is this inverter, B is this inverter. So delay for A is (g*h+p)xt and since inverter has a logical effort of 1, since they are 1X, 1X, so the output capacitance, input capacitance are same, so h is 1, so I have (1x1+p)xt, that means Tau is (A_delay)/(1+p). Now I take this circuit for the other one. C delay is similarly (gxh + p) xt, since it is 1X driving 2X, we say it is (1x2).

Please remember the capacitance here will be different, h will be larger because this is 2 times the capacitance, this is only 1 times, so h for this seek element is 2, so (1x2 + p) xt, so C delay is

(2 + p) (A delay)/(1 + p). Now if we then use those 2 equations, A delay and C delay, we say p = (2A delay - C delay) / (C delay - A delay). So I can monitor by monitoring actual delays in a normal circuit and then evaluate the parasitic delay.

(Refer Slide Time: 25:11)



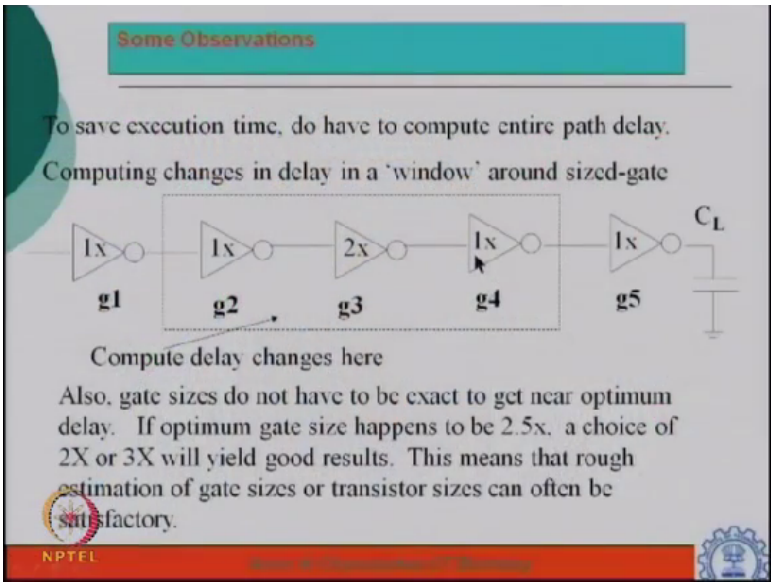
Sorry, this slide should have come earlier but does not matter. I will just repeat, this is a case of a Nand gate and we can see how the logical effort of Nand gate is different. This is my template gate this inverter which has W=1, W=2 and therefore g=1 and now I am having Nand gate in which 2 inputs or 2 N channel devices are in series and 2 p channel devices are in parallel. Since 2p channel devices are in parallel.

And we need at least W=2 for either of the reaching from VDD to the output, so each can be only 2 because then it can provide your path either A or B, worse case will at least be 2; therefore, I have a path to reach here with 2 itself is sufficient, whereas for N channel transistor, make this equal to 1. The 2N channel transistor should have double the size 2, 2 and obviously I repeat the logical effort essentially is the input capacitance of a Nand gate to the input capacitance of the inverter or template gate.

So right now I see for any input, I have 2 here, 2 here, so 4 and for the template, it is 2+1=3, so the logical effort per input of a Nand gate is 4/3; Similarly, for B, it will be 4/3, so the net will be

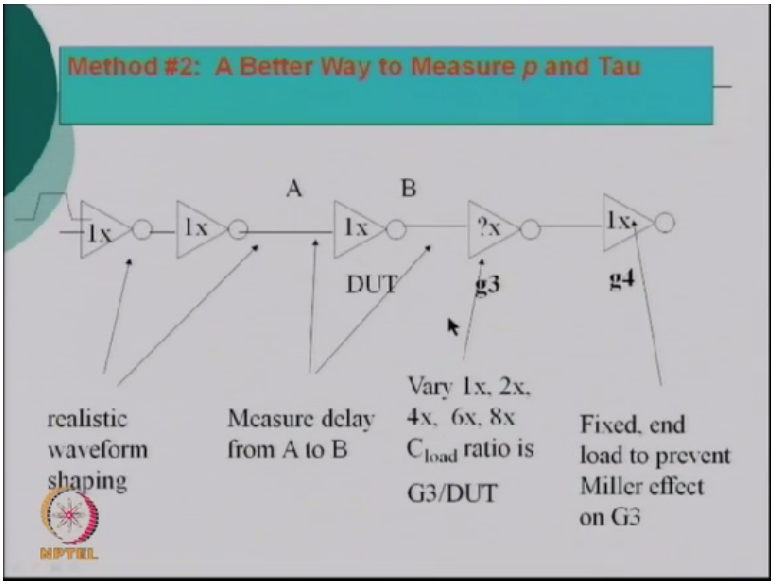
8/3 for 2 input Nand gate. Now many a times, how many times I should do this to save execution time, we may actually do 1 time of set of this kind of calculations, compute delay changes.

(Refer Slide Time: 26:47)



And then we need not repeat again and again.

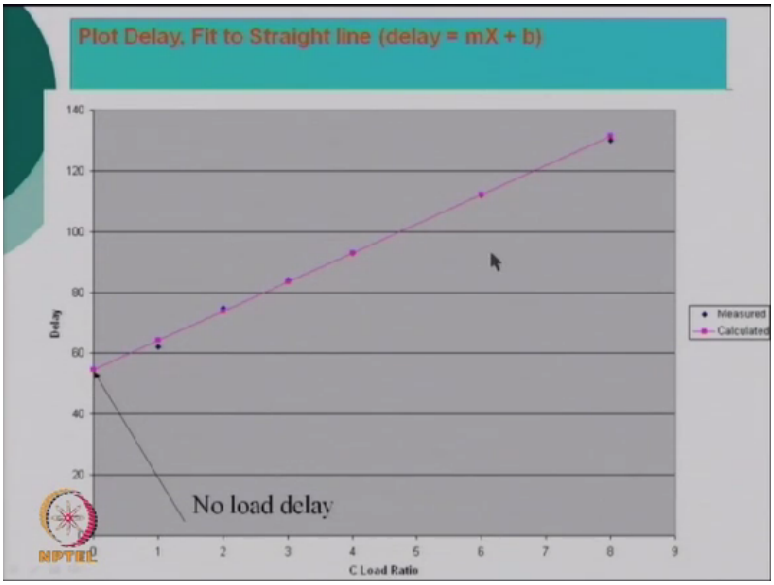
(Refer Slide Time: 26:51)



So here is a method 2, a better way to measure p and Tau, is something like this. I have 1X inverter driving 1X, then I have 1X inverter driving unknown X, okay, and there is another last current is 1X again. So I will do is, assuming that through this we get a realistic waveform because there will be delays associated with 2 inverters, so there is something we get some wave shaping as if equivalently done and you will get some realistic waveform shaping here.

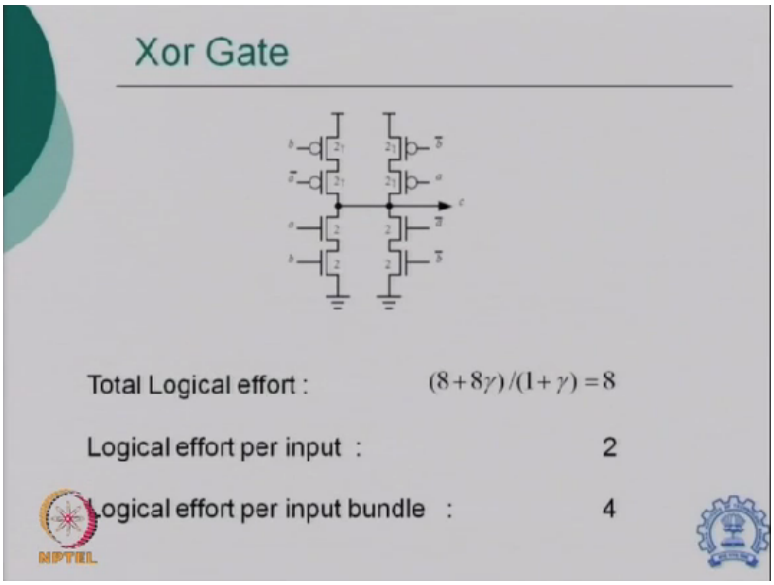
Having received an input here, I say it passes through 1X which we call is the device under test and we keep varying this G3 gate, this of course the output is fixed, so no miller effect should be allowed through G3, so we fix this and now we only vary 1X, 2X, 4X, 6X, 8X so that the C load ratio is G3/DUT, okay. Whatever is the capacitance here divided buy capacitance here, so we keep finding out that ratio and once I have number of such measurements done.

(Refer Slide Time: 28:03)



I can then plot this versus delay, number of capacity ratio we can say and once can start looking for 1X, 2X, 4X kind of thing. So when there is no load delay, that is the delay and if we have the load value increasing to the inverter size of 1X, 2X, 4X, 8X, the delay will start rising and from this, I can then evaluate by extrapolation, I can evaluate p, I can evaluate Tau and therefore because it is a straight line, mX+b, so I will be able to derive both p and Tau in one go.

(Refer Slide Time: 28:41)



Continuing with our effort once I know how to calculate G, I know how to calculate h, I also know how to know p, so obviously I now roughly know what is F which is GH and how much is p, but let us look how to evaluate G for number of gates of different kind and what is there logical effort. So here is another interesting circuit, all of you are aware that an Xor gate essentially is consist of function which is AB bar plus A bar B.

So to implement it, you need 2 inverters to create A and A bar which are essentially present, and then you need 2 N gates, A bar B and AB bar and Nand/Nor gate, so to make AB bar plus A bar B as their function. So obviously if you see an Nand gate, you need as many transistors for that, number of inputs and for a CMOS, each will require 4 and for Ring you will require further 4 of them, so by using this kind of hardware which I have shown here.

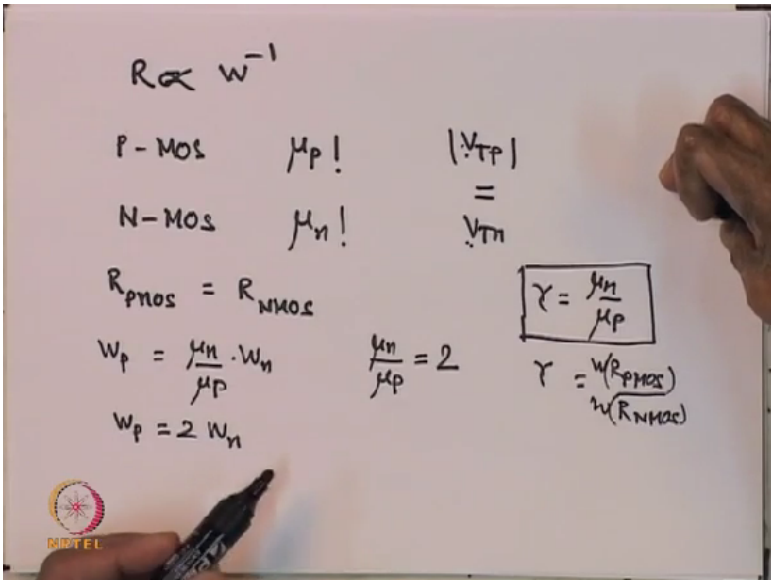
I will be able to reduce the number of transistors from let us say 12 to 8 only. So let us see what is the way circuit is functioning. There are 2 N channel devices here and 2 P channel devices here. There is another arm whose outputs at the gate, both outputs are connected as the output of the Xor and below is your another 2 N channel transistor. To make it this, the first 2 have inputs A and B and the other is A bar and B bar. On the p channel, we just do the otherwise.

We keep B as it is only on the upper side now. So it is A bar B and it is AB bar here. Okay A is lower, A must be connected both sides, B is other side, B either going to VDD or going to the

ground. By using this simple method, we will start looking into values now, since there are 2 p channel devices in either case. Now this term gamma which so far I have not discussed, now may be I will tell you what is my gamma.

The gamma essentially if you see the resistance of anything.

(Refer Slide Time: 31:04)

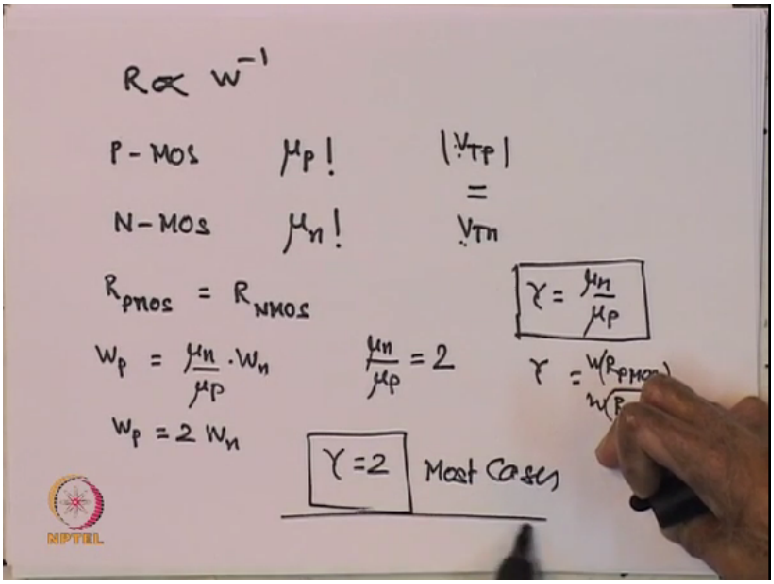


Resistance is essentially proportional to rather width to the power -1 of the transistor. So once can see from here that if you see p channel device, P-MOS, in your current equation, you get a term Mu p and in case of N-MOS, you get a term which is for N channel mobility. Right now I assume VTP and VTN are equal. If they are not equal, there may be another issue. So if you see the ratio of resistance in the 2 case and if I want R P-MOS be same as R N-MOS.

So that my timings are both side equal transient of rise and fall is same, then P channel width must be in a ratio of Mu N plus Mu P into Wn and it is typically, use Mu N/Mu P as 2 in most surface mobility ratio, typically is around 2. So we see Wp is twice Wn. That is number 2. So we say gamma is essentially Mn/Mp ratio. However, this was under an assumption that VTP is same as VTN. If VTPs are not same as VTN, gamma will be a ratio of R width related to P-MOS/width related to R N-MOS.

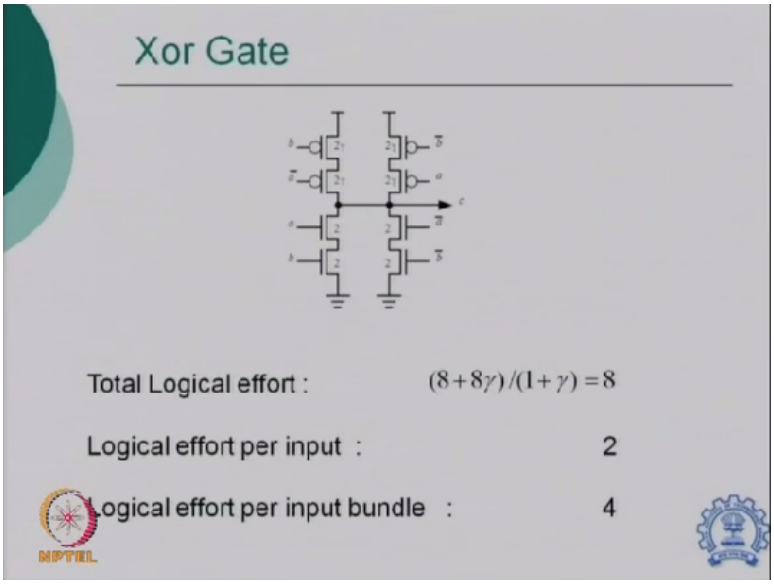
And then it will be taking care of both variations in thresholds as well as mobilities and it need not be exactly equal 2. So for a generalized case instead of using gamma as 2, we can have more generalized term which is called gamma which normally can be used 2 but need not be if p channel, N channel devices are not identical in their thresholds and their mobility ratio is also not 2.

(Refer Slide Time: 33:33)



Typically, I repeat gamma is 2 for most cases in all over designs; however, this is not true all the time, okay. So due to this, we come back to our this. So look at the Xor now.

(Refer Slide Time: 33:54)

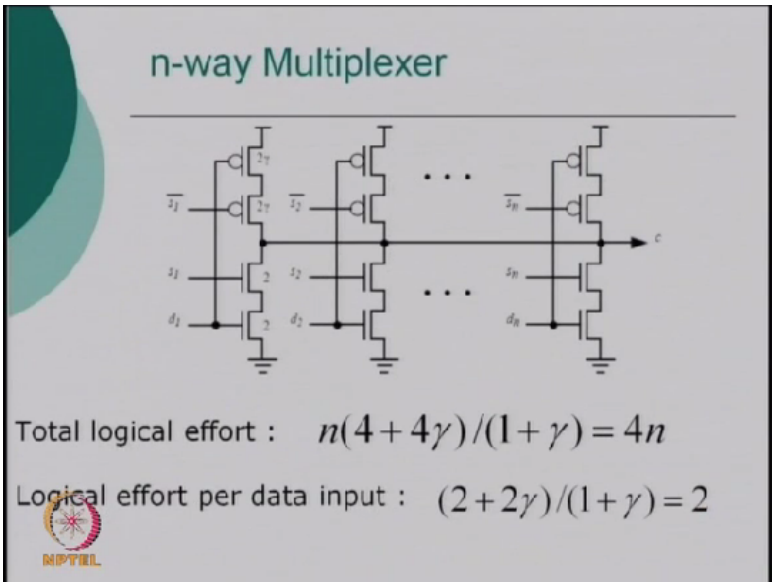


Since, there are 2 p channel devices, each should have normally we need a equivalent of 1 gamma there, so we say, to make series combination equal to 1 gamma, each of them must have a size of proportion to 2 gamma and 2 gamma here. Similarly, these 2 p channel chain showing a 2 gamma and 2 gamma in series which will give series combination of gamma. Whereas here I need for N channel, it is W proportion to 1 or series of this should be 1.

So I see to make series 1, I must have 2, 2 here, same way series combination for N channel here should have 2, 2. So now I see once I put these 2, 2 and 2 gamma, 2 gamma, 2 gamma, then I can evaluate equivalent effort compared with our template inverter, we can see for here, for input A, the logical effort per input is 2+2 gamma for A or this, 2 plus 2 gamma divided by 1+ gamma which is 2, so you have a logical effort per input is 2.

Then, we define bundle A and A bar are identical in some sense because they are complement to each other. So we say logical effort for A and A bar together is 2+2=4, since there are 2 more inputs, B and B bar and therefore bundle is BB bar which also will have the identical logical effort of 4, so we say the net logical effort is 4+4=8 which can be proved from here directly 2+2+2+2, so it is 8, (8+8 gamma)/(1+gamma) =8 and therefore logical effort of Xor gate is very high which is 8/3.

(Refer Slide Time: 35:55)



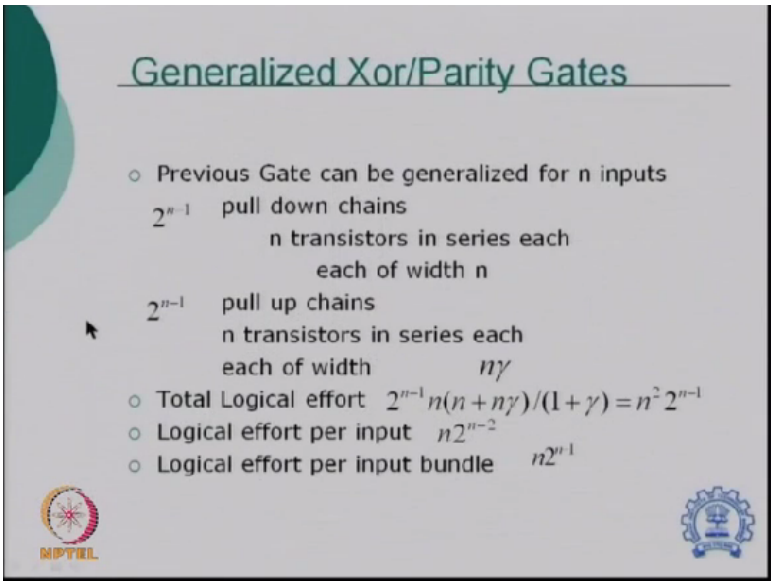
Look at this another very popular combination circuit is multiplexer, a typical multiplexer chain is shown here, it is N wave multiplexer 1, 2, 3, 4, n of them. This is a first multiplexer chain shown here, has a control input of S1, S1 bar is 2/1 and therefore this is S1, S1 bar is selecting the 2 and this is your data flowing through this. The p channel and N channel has data D1, similarly there is a data D2, then there is a data D3 and all the outputs are connected.

So we will like to see first D1 going then D2 going then Dn going as select S1, S2, S3, S4, Sn, Sn bar are actually impressed. Now let us see what I am talking. I am now saying that the central 2 transistor N channel and p channel series combination, the point center is their output, is now driven by select signal S1 and S1 bar whereas p channel and N channel forms a standard inverter, C-MOS inverter has an input D1.

Since there are 2 in series p channels, so to make it equivalent template timings. So 2 gamma and 2 gamma for N channel, it will be 2+2. So if you see per chain, per this vertical chain, you can say 4+4 gamma divided by equivalent of standard gate is 1+gamma, so 4+4gamma/1+gamma=4 and if there are n such chains, multiplied by n, this into this into this into this, the total logical effort is n times 4+4 gamma/1+gamma or 4n.

If you are interested to know per input, you can see 2 per input is D1, so it is 2+2 gamma/1+gamma and therefore it is 2 only.

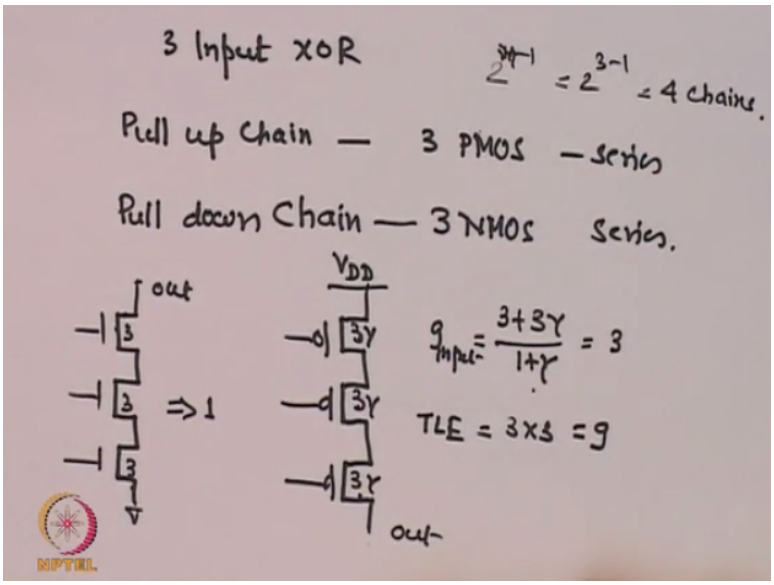
(Refer Slide Time: 37:55)



The another interesting circuit which we use in the combinational block which is essentially used in many communication circuits, essentially is the creation of parity blocks or we say detection of parity or one may say may be parity, it can be generated. Now previous gate can be generalized like the Xor gate I have shown, we know parity gate is essentially series or Xor. So if there are previous gate can be generalized for n inputs.

We say 2 to the power n-1 pull down chains, if there are n transistors in series, each of width n. Then it will be 2 to the power n-1 pull up chains n transistors in series each and has a width of p channel device, pull up will have width of n gamma. Now the total logical effort will be equal 2 to the power n-1. Now I can show you how I do it.

(Refer Slide Time: 38:56)

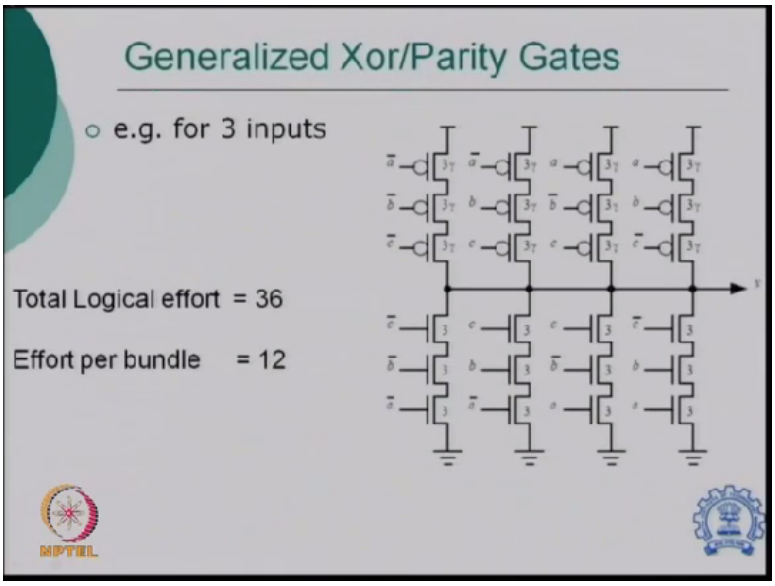


Let us take, I have the circuit which has 3 input Xor, okay. Then we can say that pull up chain that is p channel chain will have at least 3 transistors, okay, so there will be 3 transistors, 3 p channel transistors. Pull down chain will also have 3 N-MOS, both in series. Now if you say pull down, means N channel, so you will have a size, it seems there are 3 in series and if I want to make equivalent of 1, so I must make if I want to make it 1, so 3, 3, and 3.

So the combination in series will become 1 which is equal to 1 and similarly the other p channels will be, this is power supply. For this, this is output, out. So for this, I must have 3 gamma, 3 gamma, 3 gamma, so that it gives you equivalent of a gamma. So once we declare this, then for the logical effort per chain, given they are in connected here, so one can see it is 3+3 gamma/1+gamma, okay.

That means it is essentially equal to 3, okay. 3+3 gamma, 3+3 gamma, 3+3 gamma divided by 1, so g input, 1 input is 3 and so there will be 3 such chains for 3 inputs, so 3x3 is total logical effort. So it is 9. Now if you see generalized parity gate which is shown here.

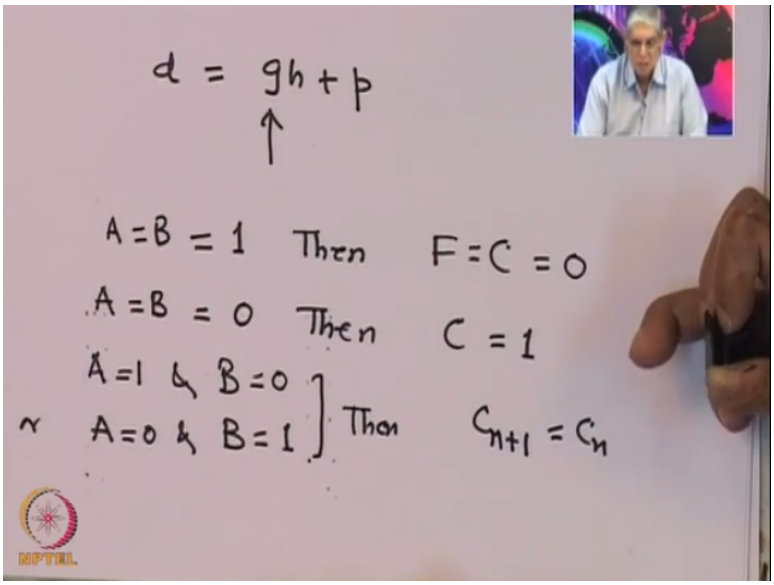
(Refer Slide Time: 41:10)



This is essentially what I was just talking. Now this figure you can see from here, if there are 3 inputs A, B, C, you will require 2 to the power n-1 chain, 2 to the power n-1, n-1 is 2 to the power 3-1, so you need 4 chains. So that is most important. So if you now calculate for individual A, B, C, A bar, B bar, C bar, A bar BC kind of each of them. The total logical effort one can say, 9 + 9 gamma / 1 + gamma which is 9 and multiplied by 4 chains, it is 36.

Effort per bundle for each input A and A bar together obviously it will be 1/3rd since there are 3 inputs, so it will be 12. So one can evaluate a logical effort g for 3 input parity gate which is shown here and one can see that the logical effort is extremely high which is 36. Now larger the logical effort, what does that essentially means. That means if the logical effort is larger, the delay is going to be larger because we already said the delay is essentially.

(Refer Slide Time: 42:33)

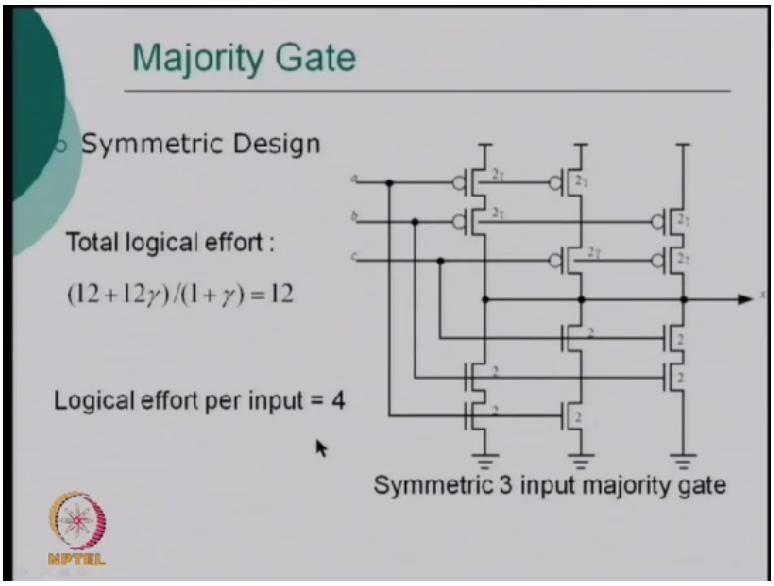


The delay not necessarily absolute delay but delay, normalized delay is gh + p. So obviously if g is larger, the delay is larger. So if you have a large chain of Xor gates to create parity, generation of parity detectors, the effort required is very, very high. However, in our all VLSR design, one of the feature which we keep looking at is how to minimize the delay or how to improve the speed.

To improve the speed, h of course is slightly not in your hand many a times because it is external to you, output loads. P of course is decided by the number of inputs we use, in this case is p is 3. So maybe since there are 3 inputs. However, g may decide, g is 36, please remember how much delay I am going to have. So I reconfigure such gates into, this is called symmetric gates. Why it is symmetric, you can see from the circuit, the upper part, this is symmetry and also this part, pull down chains are exactly identical.

Now I want to see, can I do little metric so that I can do asymmetric connections and still implement the same logic, okay.

(Refer Slide Time: 43:53)



I will come back to it little later, but I just say that can I reduce this g itself from the same block by reconfiguring the hardware itself. The another gate of interest which is uses in many combinational blocks is Majority Gate. Now what is a Majority Gate. If the majority among the input is high, the output is high. If majority among the inputs is 0, the output is 0, is called a Majority Gate. A typical Majority Gate, symmetric gate is kept here.

You have 2 N channels here corresponding 2 p channels here, you have 2 N channels here and 2 p channels here, 2 N channels here and 2 p channels here and any series combination for making it one should have 2, 2. Any series combination to have gamma, you should have both p channels in series, should be 2 gamma and I give 3 inputs, A, B, C as to fish the majority, I have to test, you can verify this whether out of when A and B are 1.

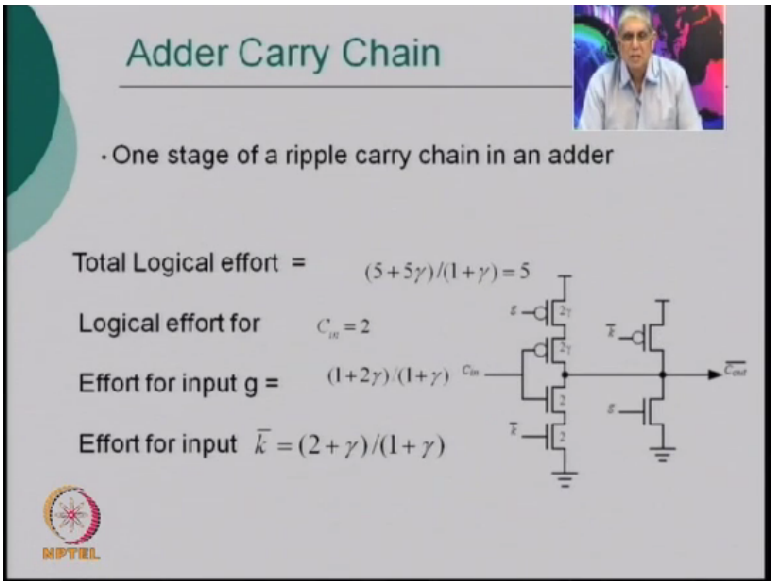
You can see from maybe one can quickly test, if A and B is 1, both these transistors are switched off, okay. C is 0, so obviously this transistor is on, this is off, okay. This is because it is receiving A, so this is off, this is on, fine, but does not matter because this is off. If you see the load on transistors since C is 0, this is off, so ground cannot be provided. However, since C is 0, this is on, B is 1, so this is off, so this chain is off.

However, if A and B are 1, these 2 N channel devices are 1, and this output is pulled down to 0. This output is pulled down to 0, so I say invert is we are obtaining, if A and B is 1, the output is

complement of what we are expecting. Whereas if B and C are 0, correspondingly 0 may appear through this chain and A and C, then will appear through this chain. So we can see if all 3 are 0s, output will remain high.

Because of either of the chains and therefore we may say it will not able to decide the output. If you see the logical effort for per input, 2+2 gamma/1+gamma which is 2. If there are 12 transistors, this 12+12 gamma/1+gamma, is 12 per input, 12/3 which is 4. So a typical Majority Gate has not very large but relatively large logical effort.

(Refer Slide Time: 46:37)

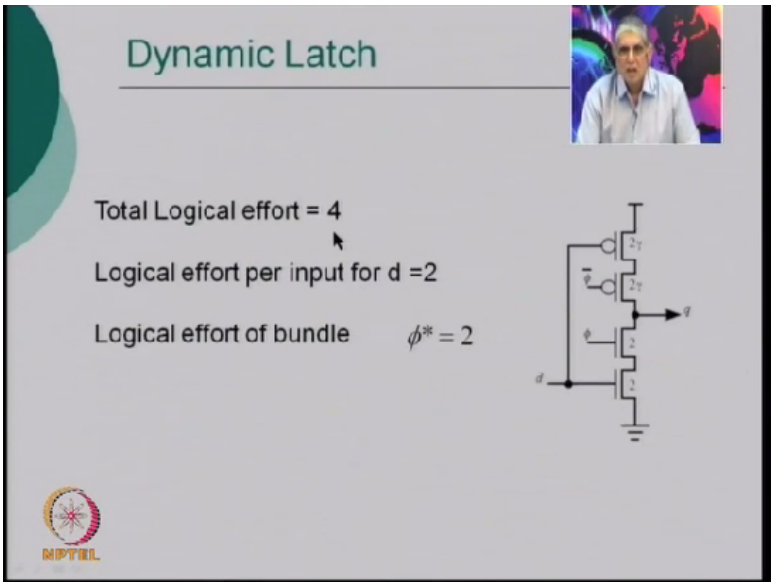


Now another circuit which is used in data parts is Adder Carry Chain, one stage of ripple carry chain is an adder. Total logical effort, you can see from carry chain if you have seen any ripple carry chain adder circuit. The carry is given between these 2 p channel devices, g and these are propagate and generate functions and one can see from this since there are 4 transistor here, 2+2 gamma/1+gamma is 2, so logical effort for input seen is 2+2 gamma is 2.

Logical effort for input g, input g is here is again 1+. Please remember now that is the catch in this, this is going to be 1, this is going to be 2 because there is only 1 inverter, so 1+2 gamma/1+gamma is the logical effort due to g. By similar argument, logical effort due to the input K, this is 2+2, so 2+gamma/1+gamma. So the net logical effort which is sum of the two is

5+5 gamma/1+gamma and therefore the logical effort of such carry chain adder, carry part on that is 5.

(Refer Slide Time: 48:05)



Another important circuit in our analysis of any sequential blocks or designing (()) (48:10) is a Latch and this is a Dynamic Latch which is essentially created due to the phi, phi bar as the clock, phi is the clock and phi bare are the clock bar and one can see here, if D is the Dynamic and phi is 1, phi bar is 0, both transistors are on. Then, the data actually as if they are shorting that and therefore only D will decide the output. If D is 1.

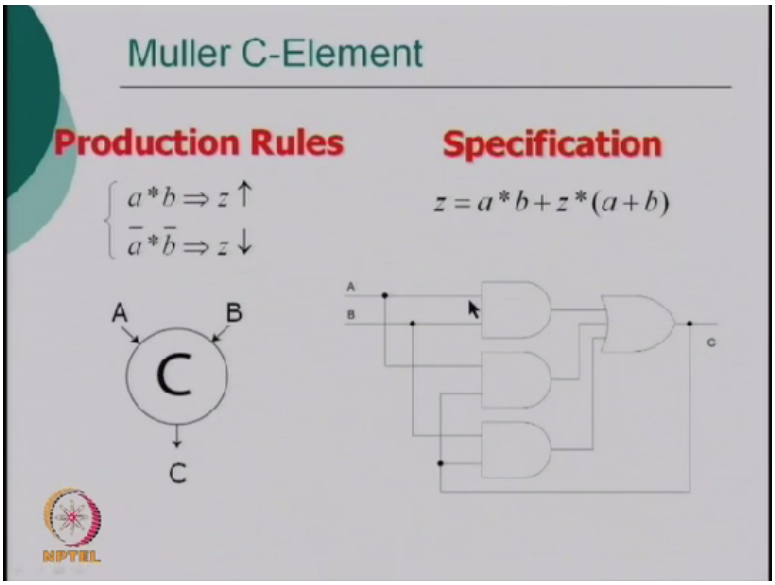
The output will go to 0. If D is 0, output will go to 1 as normal. However, if phi is 0 and phi bar is 1, which is clocked 0, then this transistor is switched off because input is 1, this transistor is switched off because input is 0, both power supply and ground lines are disconnected from the output and therefore, it retains whatever is the last state, which is what latch wants. So this is essentially interesting Dynamic Latch which is shown here.

One can see from here the logical effort, since there are series combinations of N and p, 2 and 2 gamma is the requirement, so the logical effort of this is 2+2 gamma upon 1+gamma which is 2 for each input; therefore, it is 2+2 is 4. Logical effort of bundle of course is 2 and therefore one can see that total logical effort is 2+2 which is 4. There is another element which is very important in the case of asynchronous blocks and I may actually show you this.

A Muller Element C essentially is something like this. Actually look at this figure correctly again and I may say what exactly I am saying now. If Muller element function can be written as something like this. If A is equal to B equal to 1, then output F or rather here F is in case of our case is C, the output C is essentially 0. I repeat if A and B are 1, the output of the latch is 0. If A is equal to B equal to 0, then C is 1. I repeat if A and B are 1, Cs output is 0.

If A and B are 0, then the output is 1. However, if A is 1 and B is 0 or A is 0 and B equal to 1, then whatever is the present state of C, will be same as last state of the C. This is essentially what latch wants. So if we have A and B as 1.

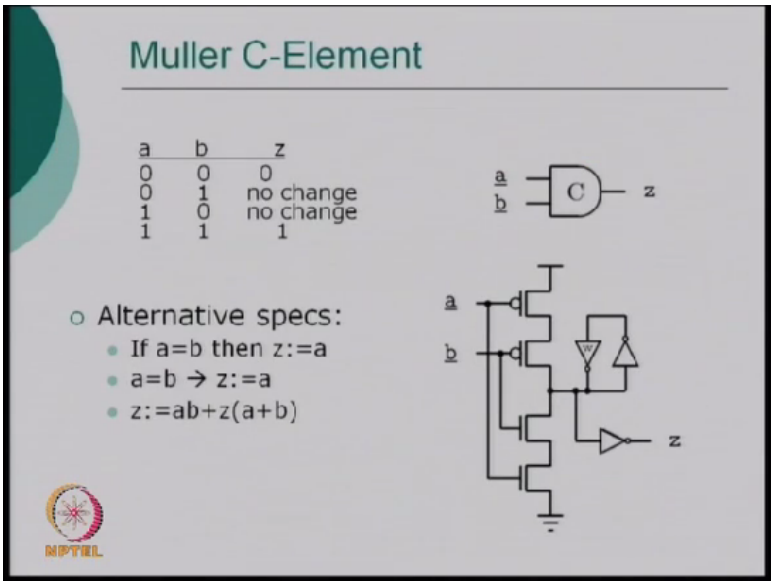
(Refer Slide Time: 51:22)



Please remember there is no clock. The data itself decides whether to latch or not to latch. So inputs are 0, it is latched to 1. If inputs are 1, it goes to 0. I can set and reset the output of a latch and it can retain the data if A is 0 and B is 1 or A is 1 and B is 0. So this essentially is, we say, is called Muller C-Element and on your right, I have shown you it is equivalents circuit as shown here, okay.

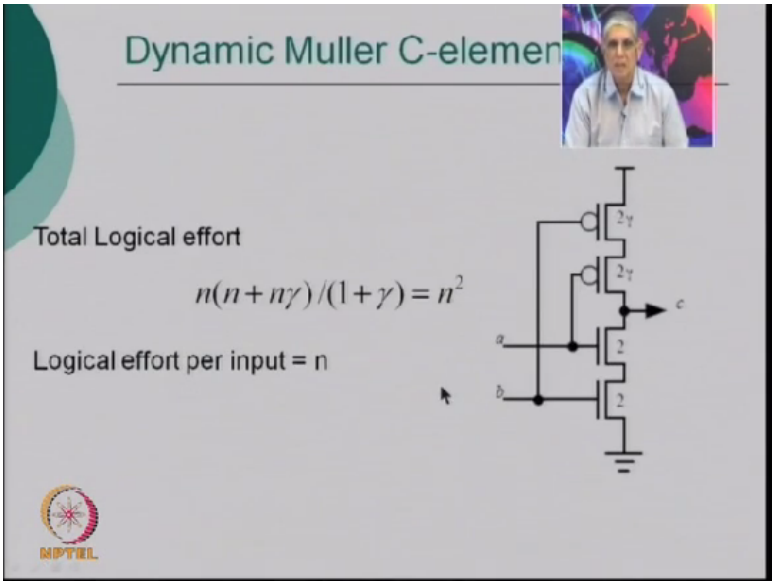
It has 3 Nand gates, 1 Nor gates, this is 2 input, 2 input, 3 Nand gates and 3 input or gates will actually create what essentially I am now talking as latch circuit, Muller C.

(Refer Slide Time: 52:11)



Well, I already said so, I am sorry I am just repeating once again and you can see another circuit possibility is this kind of equivalence which I have shown you from the schematic to this. You have 2 N channel, p channel devices. There is 1 inverter going out and this is of course one can say retaining the data. This is like a latch action from here. This is equivalently saying that.

(Refer Slide Time: 52:36)



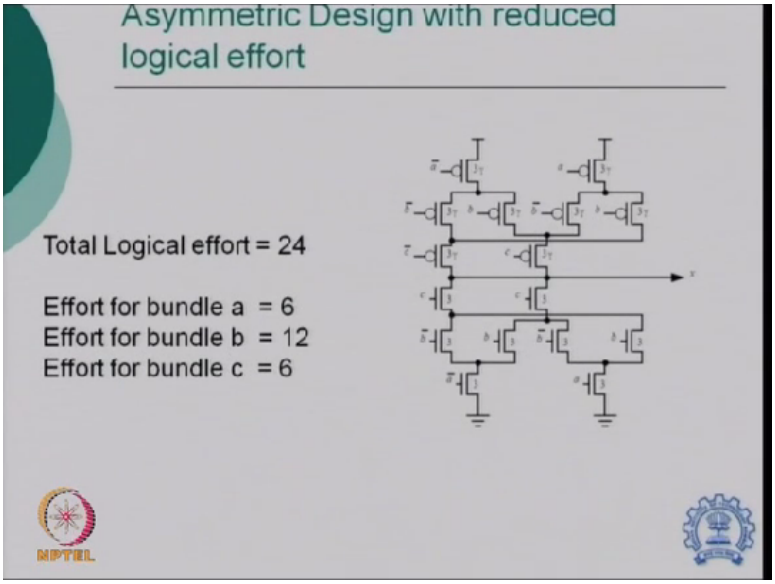
So if I see a typical Muller C-Element requirement, you have 4 transistors in series again, like phi, phi bar, you have 1 input connected to it and the other input is connected to B and now you can see if A is 1, this chain will operate, A is 1, this chain will operate, but if B is 1, this will go to 0 because this will operate and this will be pulled down to 0. If A is 1 and B is 0, in that case, A is 1 means this is operating, B is 0 that means this is not operating but this is operating.

So one will be transferred. So this is essentially Muller C, so one can see Muller C is a latch which is used without a clock and therefore extensively used in most of the asynchronous circuit which is one other area of great research in the digital design because once you say that you have asynchronous circuit. Please remember asynchronous circuit is not always the only data driven, it can be also clocked driven.

But may not be synchronous to the local clock which can be therefore a slower clock or A and B could be slower. In that case since the frequency, relative frequency is smaller than the synchronous clocks then we will say the power will be minimized in most case. So asynchronous circuits are normally preferred where power down is required; therefore, there is a method of designing a circuit now a days.

In a larger system, can be designed into 2 parts or 2 kinds of circuits, synchronous and asynchronous and general trend is to say globally asynchronous and locally synchronous. Okay GALS as the word goes since GALS will require latches in asynchronous part, so I just showed you a Dynamic Muller C-Element which we can evaluate it is logical effort and if there is an N type of this an input, N+N gamma of this and in a number of inputs, then you can say it is N square, so all logical effort per input is N.

(Refer Slide Time: 54:52)



Now this is what I was telling, I have brought it little late but does not matter. I have a typical circuit which I showed input 2 chains, circuit I have shown you. If you see earlier one has a 36, you have 3 input Xor and now I can reorganize 3 input Xor to give us what we call as lower logical effort, 3 input Xor has a logical effort of 36. Now I will like to show you that the way I organize this, okay, since something which is not related to one of the kind of parts.

One can actually remove them from the common points. So what we do is, one can see here, of course this is C bar, please check it, this is C bar and not C, so one of the chains in Xor is A bar, B bar, C bar in series, so they are connected in series and brought to the output and correspondingly it is A bar, B bar, C bar is coming from the top, okay. The second chain one can see from here A bar, B, C going to the output.

Then one can see from here A bar, B and C coming from this side to this output. If you see the other chain, you have A, B bar, C and then you can say A, B bar, C. The fourth chain is A, B, C, and A, B, C. Since you have a logical effort of 36 there, okay, logical effort of 36, 4x9, 4 chain each of them is 9, 3+3 gamma/1+gamma is 3. There are 3 such inputs of 9 and since there are 4 vertical chains, so 36 was the requirement there, logical effort.

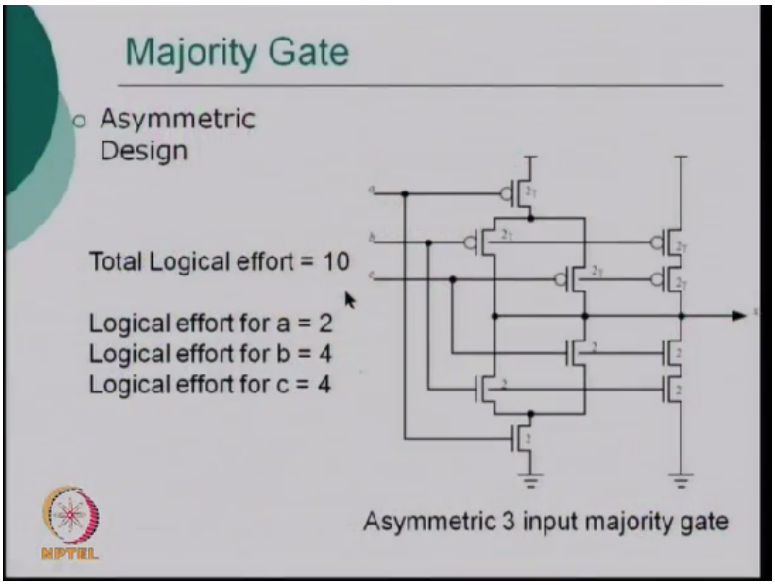
Now we see by actually separating the two, you know this point is common only to these two circuits, C bar coming from here, sorry this is also C bar, C bar coming from here, C coming from here. Now you can see for the single chain, you have 3+3 gamma/1+gamma is 3. You can see now you have, please remember 3 here, 3 here, so for each bundle, you have for effort of bundle.

So you know A is not required everywhere, you can see for A bar, only this chain is required unnecessarily you do not have to go through A bar on this chain, the idea is if those parts do not contribute to the output for A kind of input to go. We may separate them in the parallel combinations slightly separately connected and by using this effort for bundle A, A means A bar and A together.

So you can get through one of this, one of this, so 3+3 only 6, whereas for B which is coming everywhere, you can see 1, 2, 3, 4 times, so it has a logical effort of B is 12, C is only coming twice, okay and therefore it has logical effort of 6, so total 6+6+12=24, so we have saved a logical effort from 36 to 24 by proper connection and since now this is not symmetric in any of them, this is called asymmetric design.

So asymmetric has an advantage over symmetric in some sense that it has a lower logical effort. Similarly, I have done it for Majority Gate, proper connections.

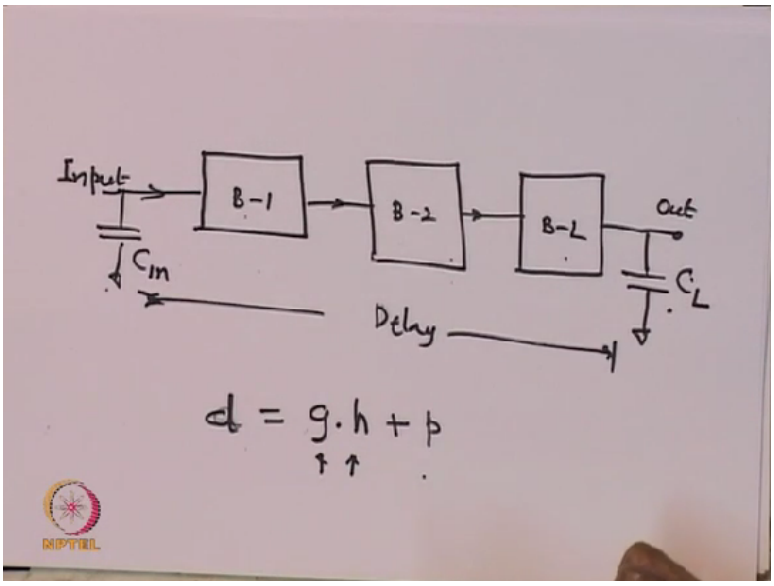
(Refer Slide Time: 58:45)



For here also, the logical effort has been minimized from 12 to 10, larger the number, it will be further reduced, can see whichever is not connected in the chain, does not appear, you can say, here there are 4 here, here there are only 3, here there are only 4. So we can now reduce not all input reaching all transistors and therefore the logical effort has become small, you can see from here, this chain is going to here and then to here, okay.

Whereas this chain is only here, whereas this chain has a 4, so we can see here the logical effort per input for A is only 2, B is 4 and C is 4, so we have a net logical effort of 10. Having seen this logical effort evaluation, at the end of the day, please note that our ultimate aim of any circuit is to evaluate the net delay.

(Refer Slide Time: 59:46)



Let us say these are blocks, data is flowing in a number of some logical blocks and there is a formal C load which is sitting here, here is an input capacitance which is C in, block 1, block 2, block 3. Now I want to see when the time taken or delay taken for input starts from here and reaches an output here, delay. Now one can see that this C in and this block cannot be changed at the input, this has an output external capacitance C L, not much can be varied here.

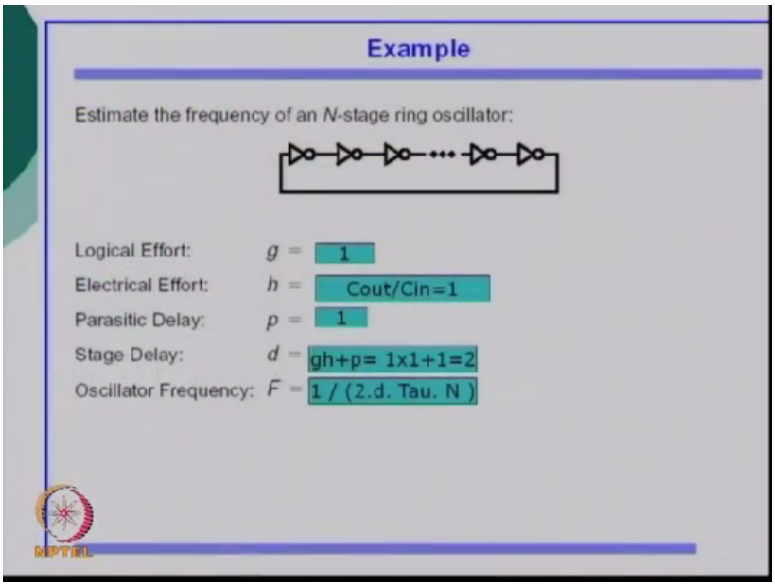
However, one can play the sizing in this and the paths in this, for example you may have a smaller number of gates in a particular path. The delay is essentially decided in a path by which is the slowest we call the critical path. So we will be able to derive to improve the critical path or to find the critical path by actually sizing number of transistors in these 2 blocks because this may not be because this is an input driver.

So using this, we will be able to actually design a circuit which has the least path delay. However, before we do this, we must now figure it out how to estimate the delay taken from here to here, if I know this delay, I can estimate for given blocks then I will be able to actually optimize the path delay and that is what essentially logical effort is, how to minimize the delay. We know delay, already I have said.

The delay is equal to g which is called a logical effort h which is called the electrical effort C L by C in plus of course the parasitic delay net. So I will like to see in each block what is the g

product and then we will say add to that p and then see what is the net h and evaluate the delay. So let us see this technique to evaluate delay. So let us take a simple ring oscillator example.

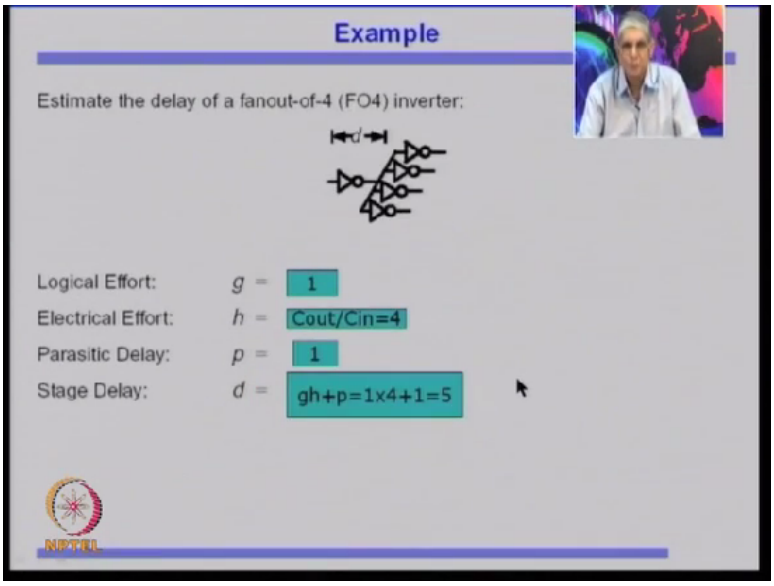
(Refer Slide Time: 01:02:18)



We are now observing estimate the frequency of an N-stage ring oscillator is my problem. Now each inverter in my chain if you see, each inverter in my chain has a logical effort of 1. We have set inverter as a logical effort g=1, okay. So g is 1. Electrical effort is essentially since the all inverter see the same equivalent inverters, so C out for each of them is same as C in, C out is same as C in.

Therefore, the electrical effort for each is C out and C in. Since it is an inverter, each inverter gives 1 parasitic delay which is this, so stage delay essentially is gh + p which is d which is 1x1+1 which is 2 and then the oscillator frequency will be 1/2d, 2 d means twice of that because you know at times 1 will go, next time 0 will go, so you need twice timing and N is the number of stages through which or 2N number of stages are equivalently required to create the maximum oscillator frequency. Of course, it depends on the even out numbers as well.

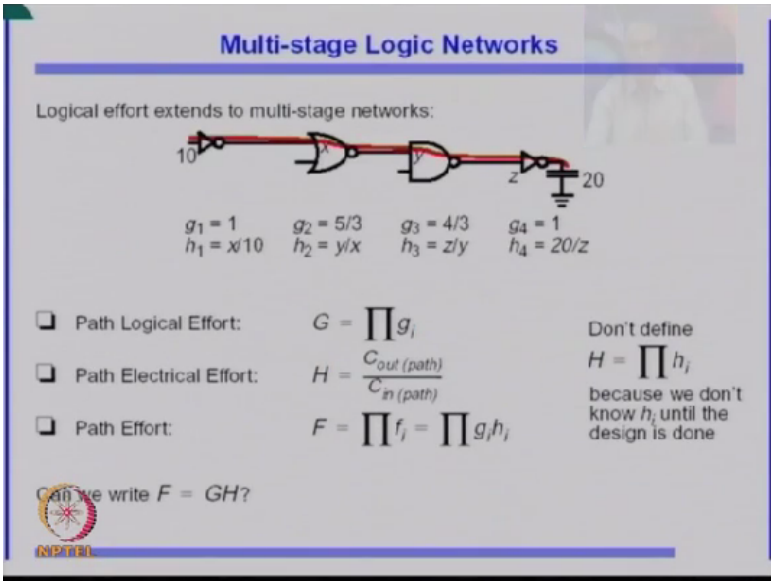
(Refer Slide Time: 01:03:34)



In all the design of a digital block, it is a common practice to design a delay for the load which is equivalent of 4 times the inverter load and such circuits are called 4 fan out inverters load or which is called FO4, okay, fanout-of-4. So here is an example, okay. This inverter is our driving inverter and this is our load together putting into 4 of them. For this inverter, I want to know what is the logical effort delay it will create.

We know logical effort of this inverter g is 1, however, now the electrical effort of each will give you 1C, 2C, 3C, 4C; input is 1C, so the C out/C in is 4. Since you have only 1 inverter here, the parasitic delay is 1, so the staged delay is gh+p=1x4+1=5. So for an FO4, an inverter has a delay of 5 for an FO4 load.

(Refer Slide Time: 01:04:41)



Now here is an interesting example which will verify what I am all the time talking about, how to get the total path delay. This circuit of course is a salt circuit, both in this Sproull Sutherland book, so I have no credits taken anywhere. Please remember most of the examples solved in my, I will give you some unsolved and maybe solve a few of them next time but at that time. Please remember this data is also taken from Sproull's book, Sproull Sutherland Harris book.

You have an inverter driving a 2 input Nor gate which one of the output of that is one of input of a 2 input Nand gate and the output of this Nand gate is inverted through inverter here and is driving a load which has a capacity load of 20 units. Let us say this inverter has a input capacitance of 10 units, okay. So now let us calculate G and H for each of them. Normally the method one we found out is we can calculate C out by C in, C out by C in, C out by C in, that is H for each of them.

However, right now the net H if you want to know how much is the electrical effort, all that we are interested in is C L divided by C in which is essentially 2 in our case. Why I am saying you because if you see h1, then it is say let us say this has a dimension of X, this has a capacitance here X, so the ratio is x/10. If it is y here, Y/X; if it is Z, Z/Y and if it is here 20/Z. So if you see then it should occur as X/10xY/XxZ/Yx20/Z should give me h which is equal to 20 by Z.

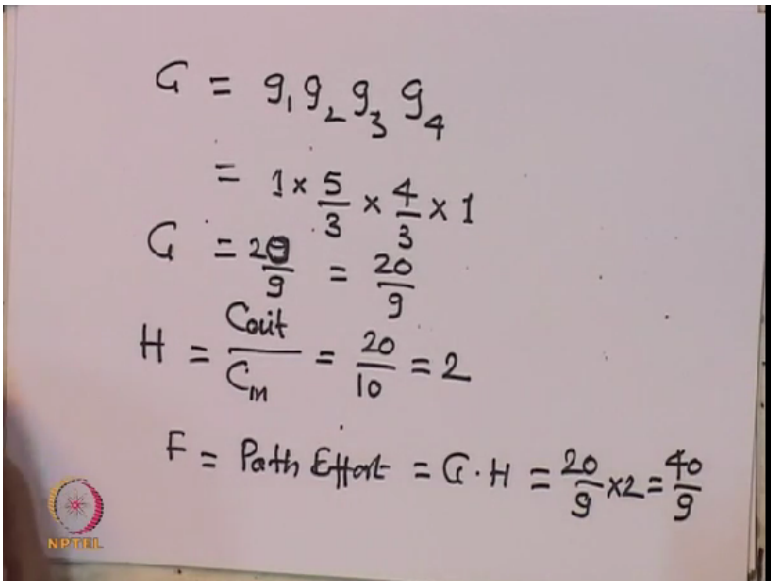
So this is what essentially you are meaning but I repeatedly telling you do not define h as product of h1, h2, h3, h4. This is not to be defined like this. H is a net path effort, is output capacitance divided by the input capacitance, C out divided C in across the path. Do not go piecewise. However, if you look at the logical effort path, this has a logical effort of g1, this has a logical effort of g2, this has a logical effort of g3, and this has logical effort of g4.

Since it is Nand inverter, it has a logical effort of 1, this is 5/3; this is a 2 input Nand, so it has a logical effort of 4/3; inverter again here. Now the way we define the net logical effort, I already said I have started with inverter as my template. Now for this, this is fair enough, so it will be 5/3 with reference to inverter but this is not receiving only inverter. So we say with reference to this for this.

You must now see 4/3 x 5/3 x 1 is the net logical effort to come here from input to the output and if you go here, further multiplied by g4. So we define the total path logical effort G is the product of g I that means g1, g2, g3, g4 in the path. Then we define path, GH is the Capital G into capital H and please as I keep saying do not do this. I keep saying, we can just write, please do not write, of course F is f1 x f2 x f3 but instead of all of them.

You actually write the net path effort F is capital G into capital H as I do not want to define H as product of h1, h2, h3. Now, let us do these calculations for the values which we gave.

(Refer Slide Time: 01:08:56)

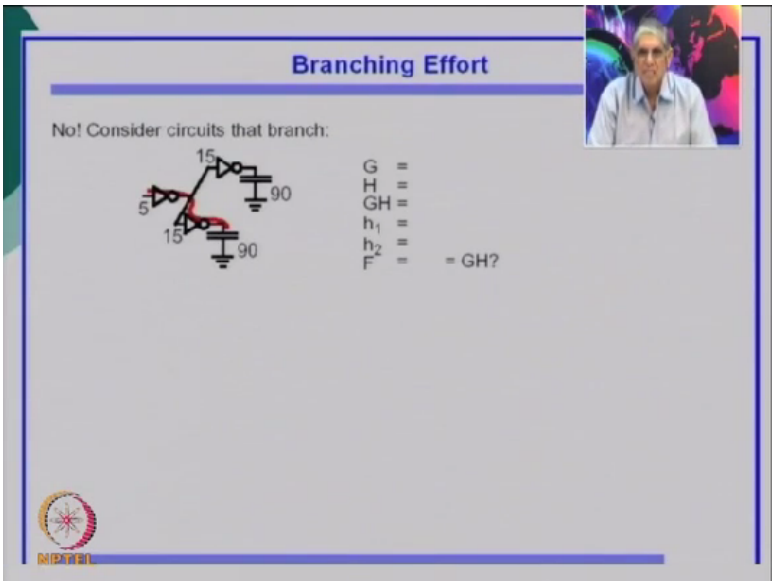


You look at it. The actual G, if you see for this circuit, the capital G is g1 g2 g3 and g4, okay, which is essentially 1 x 5/3 x 4/3 x 1 which is essentially equal to 20/9, is my capital G, 20/9, okay. If you see the capital H which is C out/C in and it is 20/10 which is 2. So F the path we can say F, path effort, this is the net path effort GH is GxH which is 20/9x2 which is 40/9. Why we want to do this.

Because if we use this expression, I have no idea actually how much is the value of x, y, z, so then I will never be evaluated h1 h2 h3 h4 product and find H and something like this, so the method I suggest is do not always evaluate h1 h2 h3 initially, you only see the capital H which is the output capacitance by input capacitance and evaluate g for the gates in the path, the red path shown is the path.

Please remember the other input also placed at the output values but the path is essentially shown as the red line. So I can therefore calculate the path effort as 40/9 for this above circuit, okay.

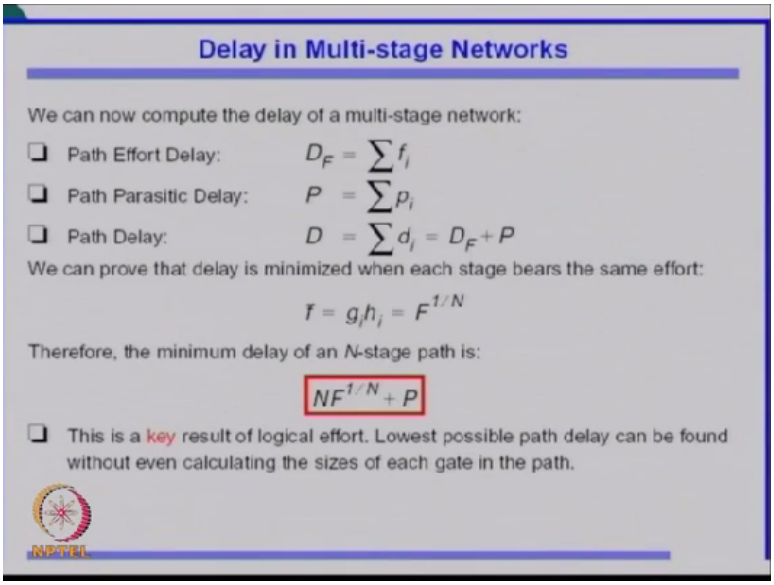
(Refer Slide Time: 01:10:52)



Now there is another interesting thing which we will see later, if there is a branch that one inverter driving the two such inverters, how do we calculate GH and once we do this, for example in this case H for this is 90/5 and this is also 90/5, though this inverter capacitance is known to me 15 by 15. So if I want to know the G for each path, this is 1x1=1 and H and 90//5 which is 18, so GH is 18 on the either path, okay.

So the effort for each path is 18 as such because capital G is 1, 1x1=1 and capital H is 90/5 which is 18, so the branching effort also each branch has F value of 18.

(Refer Slide Time: 01:11:46)



I think before we start ahead of this, I think we will stop today here and we will continue from here next time and then I think we will know go on a little faster on the rest of the circuit having explained you the basic principal of actually getting the delay across the path, how to evaluate G, how to evaluate H and once we do this, we will be able to quickly figure out the net delay and alternatively if we know the multi-path delay, we will be able to calculate individual h values.

And therefore, individual stage delays and once we get that. We may be able to actually design the input capacitance of each block in the path and that is what essentially the logical effort is going to do.