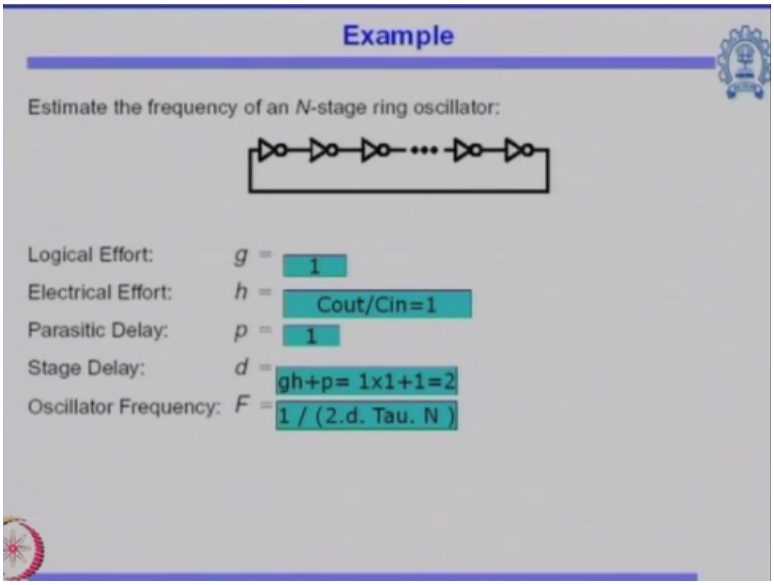
Advanced VLSI Design
Prof. A. N. Chandorkar
Department of Electrical Engineering
Indian Institute of Technology- Bombay

Lecture – 05
Logical Effort - A way of Designing Fast CMOS Circuits -Part III

We have been discussing the design of a fast CMOS circuit and the technique we have been discussing so far is logical effort. We will continue with that thing and today probably we will finally finish off what is exactly the word logical effort and its importance in the designing circuits. Last time we have discussed about calculating the logical efforts and also electrical effort. Today, we shall look into what we call as the net result of all the effort which is what we call delay estimation.

(Refer Slide Time: 00:53)

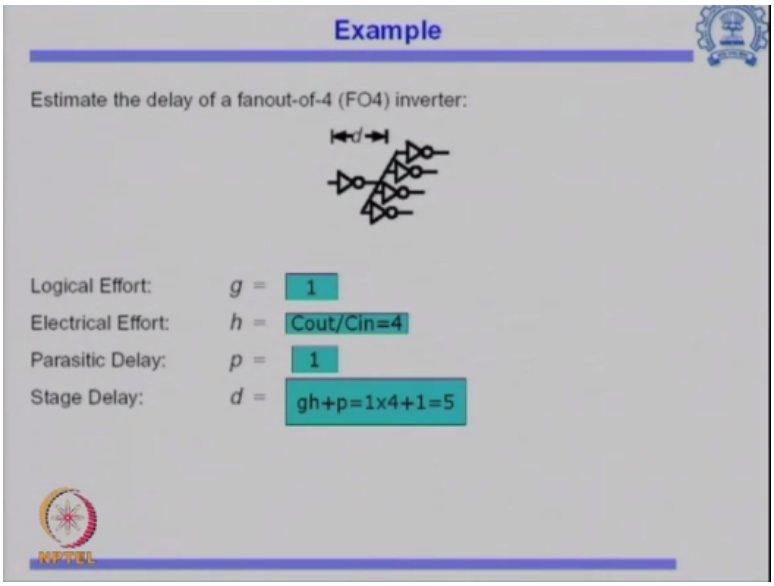


So here is a simple circuit shown to you, it is a ring oscillator, N-stage ring oscillator and one wants to know what is the delay essentially or frequency related to this ring counter. Since all are inverter, individual inverter has a logical effort of one, the electrical effort since all inverters are identical their input capacitance which is going to be the output of the next stage is always same as for all the inverters and therefore Cout by seen in this case is 1.

The Parasitic Delay since it is an inverter, it has a Parasitic Delay of 1 unit and therefore the Stage Delay D is gh + p which is equal to 1 x 1 + 1 which is 2 and as we know upon delay of

course since it is a 1.0 transmission for oscillator so we say it is 1/2.d. Tau. x N as the oscillator frequency.

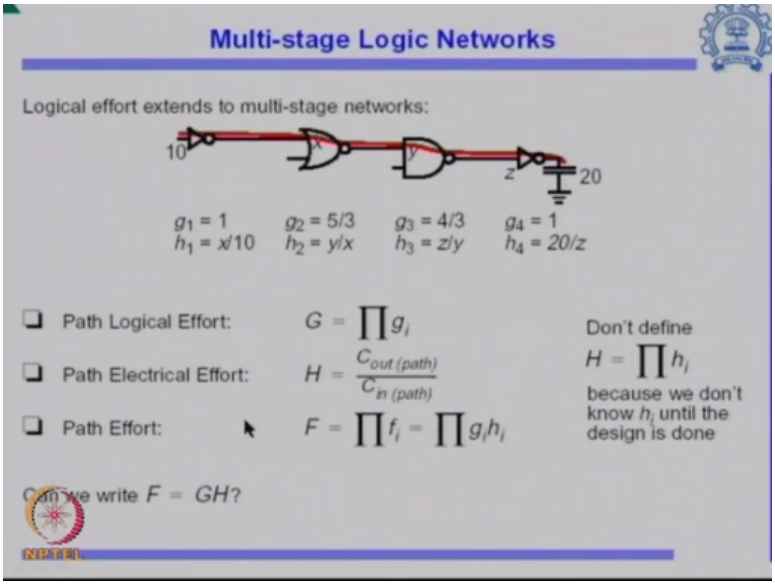
(Refer Slide Time: 01:51)



There is something interesting about the circuit as I last time said, there are all the circuits normally designed prior to the actual load knowledge we say it is fanout-of-4 inverter design essentially means an inverter is driving equivalent of four inverters load which is shown here in the figure. There are four inverters and driven by one single inverter. If you look at the logical effort for this inverter than the g =1 the electrical effort is since there are four inverters.

So the Cout is four times Cin, so essentially the HEC out by seen as 4. The Parasitic Delay of this inverter being 1, therefore the Stage Delay is gh+p which is 1x4+1 which is 5 units.

(Refer Slide Time: 02:45)



So in this fashion we will be able to derive the stage delay and let us calculate for a larger circuit which essentially say there are multiple stages logic networks, an example shown here, one can see here that there is an inverter whose input capacitance is 10 units and its output capacitance essentially for the whole network is 20 units of capacitance. Now since this inverter is driving an NOR gate and one of the input is driven by it.

Let us say it has an input capacitance proportion to X. Similarly, this NOR gate is driving another NAND gate, one of the input of a NOR gate, NAND gate and let us say it has an input capacitors proportion to Y and then finally this NAND gate drives a buffer stage or inverter which input capacitance is Z and the output load of coil I say is 20 units. Now I want to find the delay for the signal starting at this input side.

And how much time it takes to charge this capacitor at the output the technique which logical effort people have suggested is the following. We first actually ascribed to each of such stages, 1, 2, 3, 4, there are four-stage circuit now, so we say for the first G1 the logical effort for inverter is one, logical effort of a NOR gate venue is 5/3 for 2 inputs and logical effort of a NAND gate 2 input is again 4/3 inverter being one so G4 is 1.

So we actually known G1, G2, G3, G4. We also can see that if you see localized electrical effort and let us say since X is the output capacitance of this, then we say X/10 is the electrical effort

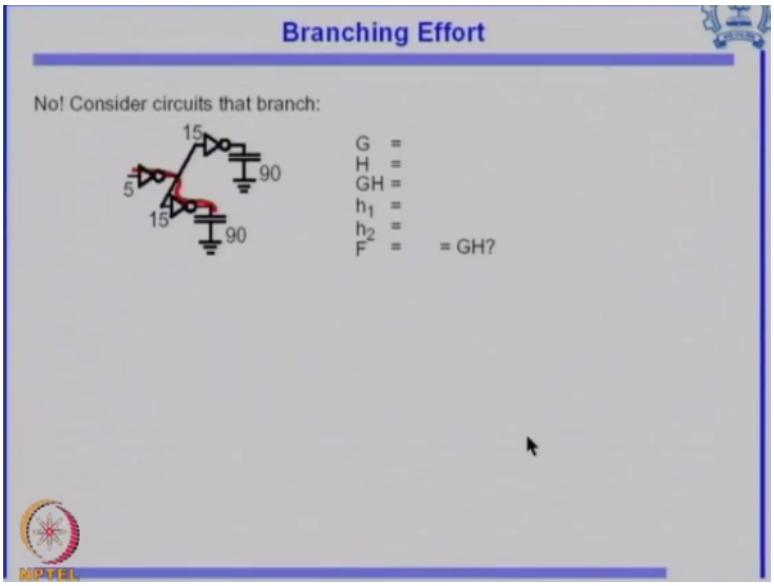
for this. Then Y/X for this, Z/Y for this and 20/Z for this. However, I repeatedly told you last time the net electrical effort should not be calculated by multiplying on this but essentially is always 20 divided by 10 which is the C output path or Cin path.

So in our case for example it is 2. So if you want to calculate the net part delay so the method is first calculate the net logical effort along the path and the method which we suggest is the net G or net pathological effort is the product of G1, G2, G3, G4. And this stem from the fact that we have actually chosen in our logical effort our template as single inverter as which with P channel device has twice the size of N channel.

However, any other circuit can act as a template and since it can act like a template with reference to that, the next logical effort can be calculated. So based on this analysis one can see as if the net pathological effort is G1, G2, G3, G4 and a net path electrical effort I keep saying it is just the output capacitance by input capacitance on this and the path effort that is total part effort F which is GH is essentially one has to actually make a product of G1 H1 into G2 H2 into G3H3 into G4H4 and then we can write the net path effort F as GH.

Please remember I repeatedly saying you H should not be calculated as product of H1, H2, H3, H4 and so on.

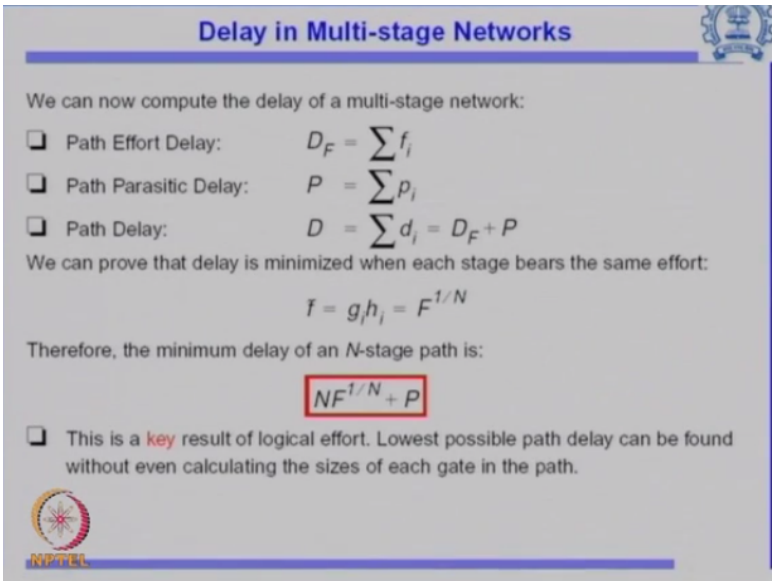
(Refer Slide Time: 06:05)



This is an example which you can like to see one inverter is driven by another two inverters, each has a capacitor of 15 and 15 and the first inverter has an input capacitance of 5 and the output capacitance both inverters driving is of 90 units. So if I want to find along this path the net G then we know G for this G1H5, G1 this for second invertor is also 1 and 1, so the net logical effort along this path is only 1.

So G capital is 1, the H is of course 90 divided by 5, so it is 18 and therefore GH is 18 into 1 is 18 and one can then also find in between H1/H2 so for let us say this is H1, so we say 15/5 is H1 and 90/15 is H2 and therefore one can always evaluate the net F which is GH which is 18 in our case.

(Refer Slide Time: 07:08)



Now to take an example little more seriously we can now compute the delay of a multistate network by saying that path effort delay we know that if I have the path effort delay defined as DF, then it is essentially the sum of all the stage efforts, F1, F2, F3 for all stages which is G1 H1 kind of thing so the path parasitic delay also by same logic one can say, some of all the parasitic delay along the path.

And please remember a path parasitic delay for inverter is 1, for 2 input any circuit is 2, 3 input it will be 3 and so on. Therefore, the delay essentially across is D is equal to DF + P RDI which is D is taken from here, so DF + P, P is the sum of that. Now one must evaluate this what we call as

DF and once we know this DF which is the path effort delay then we will be able to calculate the net D.

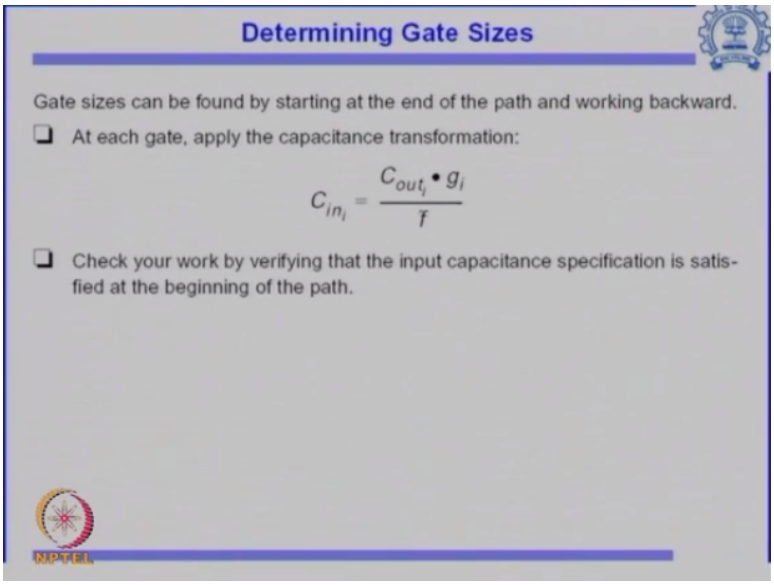
So how do we calculate, we say that by theory one can proven this, I am not proving it here, you can look into the book's theory there. What Sproull and Sutherland and Harris have suggested and that can be proved rather easily that if you want the delay to be minimized the best way of doing is that each stage of the path of this network should have same stage delay that is the f bar as I wrote here must be equal to G1 H1 equal to G2 H2 equal to G3 H3 equal to G3 H3.

And since f is nothing but Fi this. So one can say f which is essentially the net path this GH so we say f to the power, see let us G1 H1 into G2 H2 into and if they are all equal then it will be GH to the power 1/N individually and therefore it is F to the power 1/N if N stages are actually being used. And therefore I substitute this part in this therefore the minimum delay of N stage path is there are N stages each has a delay of F to the power 1/N + the net parasitic delay.

So, we say it is NF to the power 1/N + P. The key result of this logical effort is exactly this that I can evaluate the net path delay if I know the number of stages, if I say they are optimized to a minimum delay circuit so I can evaluate F which is nothing but GH and since than as a know as the stages I go through number of blocks there, I can evaluate the net P, P1 + P2, + P3 or PN and once I know this I know the path delay.

What is the advantage of this, since I have already said for optimal there are stage delays known which can be found from F actually and since F is known, 1/N is known, I know optimal delay optimal stage effort GH and if I know G for that, I can then evaluate H for every stage, that is exactly H means the ratio capacitance or therefore the ratio of sizes the two sides of thin gate.

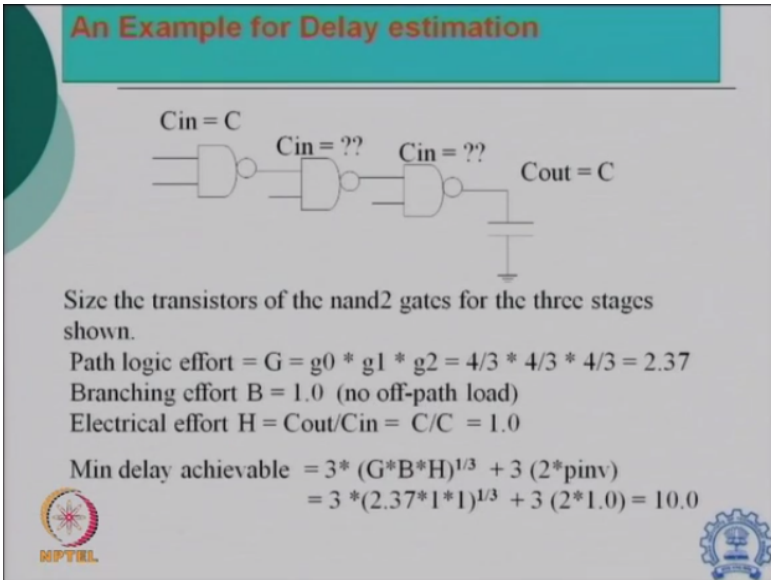
(Refer Slide Time: 10:28)



So here is a problem which we can then take care in our next analysis which is what we are just now, at each gait applied the capacitance transformer, therefore since GH equal is common to all which is equal to F bar, so Cin 1 is equal to Cout 2 by GY by F bar and since I know for each gait what is G, I know what is the output load I have, I know the stage effort because I know the net F.

And therefore I can evaluate the last Cin 1 of that stage and by keep doing this back calculating until the first inverter, the first stage is obtained.

(Refer Slide Time: 10:59)



I will give an example, let us say there are NAND gates, for simplicity the assumption is all gates

have identical structures and there are two inputs only, so we can see from here the first stage Cin is known to me, if this is my signal where I am actually putting and Cin is C, the next NAND gate which is driven by this, I do not know what is it is Cin. The second gait is driving another NAND gait.

I also do not know what is it is Cin, but I know what is the Cout and I write now assume that the output capacitance to be driven is same and what input capacitance is. Now I want to know the value of Cin here, value of Cin here and I want to, Cin means essentially I know the width because capacitance proportion to width, so I can size the transistors here and here. Now let us say size is down to NAND gait for three stage shown here, the path logic effort.

What is the net path logic, there are three stages, so G0 into G1 into G3, since all of them are NAND gates, each NAND gate has a logical effort of 4/3, so 4/3 into 4/3 into 4/3. If I do a calculation this turns out to be 2.37. Now this word right now I assume that since each gait is only driving one input but there is a possibility that this would have driven another input, another gait here as found out.

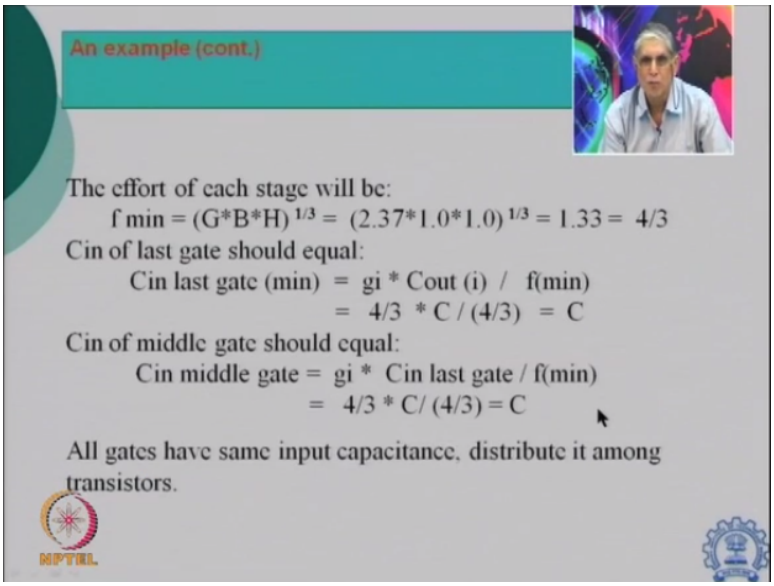
But right now we say there is no branches, individually one each is going through in the path, so we say branch effort is, when we will come back to this later again, no branch path in this, so B is 1. And then we say electrical effort H is Cout / Cin which is essentially C/C which 1, so if we say GBH that is GBH product than this is the path effort and the path effort is therefore 2.37. We have just said that the minimum delay occurs when each stage has same stage effort.

And using that analysis, we derived that the minimum delay achievable between input to the output is N which is 3, one two three into GBH which is F to the power 1/N which is 1/3 + each stage delay since there are two input NAND gates, each has two parasitic delays and since there are three of them, one can say 3 into 2 being worth as the formulation. Substituting the value, 3 into 3 into 2.37 into 1 into 1 to the power 1/3 + 3 into 2.

Because inverter has a delay 1, two of them, so it is 6, 6 + 4 which means 10, so the minimum delay achievable to us is 10 for this circuit and once I know this, now the net path delay and

therefore, I know f, since F is equal to F to the power 1/3 is F, GBH and therefore I can I will be able to now evaluate from there the capacitance back.

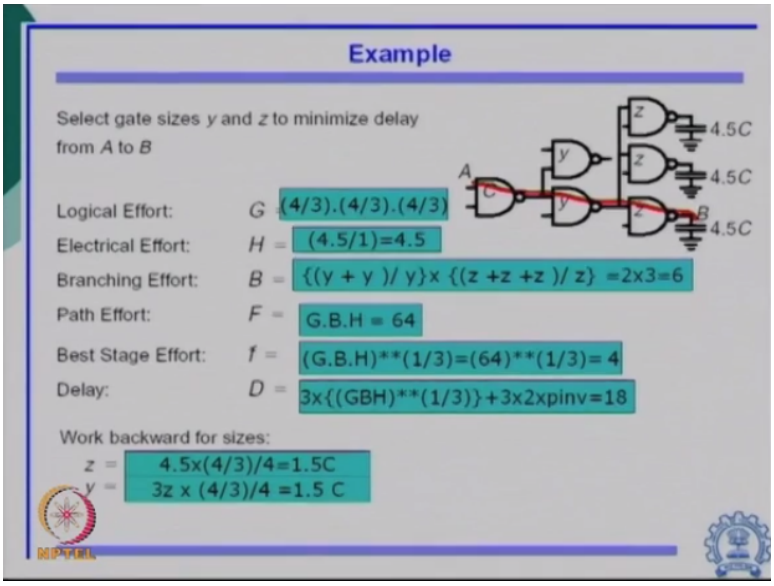
(Refer Slide Time: 14:11)



The effort of each stage which we call the optimal f minimum or f bar is equal to GBH to the power 1/3 and if I do it this occurs to be 4/3. Cin last gate should be equal to, Cin in the last gate minimum is equal GI into Cout I/f minimum as to be just derives, so it is 4/3 into C divided by 4/3 so C. So what we now say that the last stage of this should have input capacitance same as the output capacitance.

Now using that for the next stage middle gate we calculate again GI Cin last gate by f minimum which is again 4/3 into C divided by 4/3 C, so it seems that for mitigating the minimum delay and the total part delay of 10, the capacitances are same across all inputs and that we could derive.

(Refer Slide Time: 15:10)



Another example so to pay it slightly different from the last one, now here is something which I am adding additionally, one can see from here in the circuit, there is a two input NAND gate, which is driving two input, two NAND gates, each of the output of this stage is not only driving this gate, but also driving this gate. This NAND gate only drives this one last NAND gait but also drives another two NAND gates, which are shown here.

So one can see output here is branch between two of them, output here in branch between three of them and therefore the net contribution coming here will not only come from here but these two, net contribution to capacitance will not only come here but also these two and we must find how much additional current or additional path delay will occur because we are putting additional path along the net path for which we are looking for a delay.

We assumed here that all the output capacitance for every stage is 4.5 C and let us calculate now A to B what is the path delay, a minimum path delay. Let us do the same analysis once again from A to B, logical effort, this is two input NAND gate, 4/3 is the first, 4/3 is the second. 4/3, 4/3, so product of 4/3, 4/3, 4/3. The electrical effort is output capacitance divided by input capacitance. Okay I forgot to show, the input capacitance here is 1, okay.

So 4.5 divided by 1, so it is 4.5. And the branch effort is essentially now total branches two here and three here should be taken care in evaluation branch effort the way we calculate is. The

actual capacitance here is Y + Y whereas we are only driving one of them so we say Y + Y divided by Y is the actually the real effort which is required to drive this because you have twice the current but only one of that will take care.

So we say Y + Y/Y into because is in series, effort is in series of that, so we say now it is Z + Z + Z, 3 Z's divided by the 1 which you are looking for one of them on the Z, so it is Z + Z +/Z. So the essential branch effort is 2 due to this and 3 due to this, so 2 into 3 is 6. And since we now know G, we now know H, we also know branch effort 6, 4/3, 4/3, 4/3, this is 4.5, this is 6. If I multiply GBH then I get F is equal to 64.

Now we know the best stage effort or the minimum stage effort a bar shown here is nothing but GBH to the power 1/3 because there are 3 stages of path, so it is GBH to the power 1/3, which is 64 to the power 1/3 which essentially means a bar or at minimum is 4. Now we know that the delay along the path is number of gates stages so 1 to 3 stages multiplied by a bar which calculatedly, we rewrote that at bar + the delay due to each gate to input.

NAND gate will give you two inverter gate equivalent delay, this will be 2 inverter 2 inverter so it is essentially 2+2+2 which is 6, this being inverter. If I substitute this value 4 here + 6 here, so I get 3 into 2, 6; + 4 into 3, 12; 12 + 6 is 18. So the path delay going from input A to input B which is minimum path delay is essentially 18 units. If I want the actual delay, then multiply it by Tau until we get absolute delay 18 Tau if Tau is known from the technology.

Now coming back to find what should be the capacitances here because that is the sizing of this gate transistor sitting here, I must evaluate the values of Z, Y, and right now I assume that Y is our same, they need not be actually same, it can be 2 Z or it can be 4 Z, even then it can be calculated, it is not my point to say that all should be equal. For simplicity example is taken where all the capacitances, all three gates are identical.

If I do so to calculate Z, you can see Cout, divided by G which is 4 x 3 because of there is 2 input NAND gate, divided by 4 because this is the f bar which I achieved is a f bar, so a repeat Cout into G of that gait which we are calculating for Cout to Cin ratio divided by at minimum which is

4, so 4.5 into 4/3 by 4 and if we calculate this it become 1.5 C. C is 1 in our case. Similarly, if once I know now Z here okay, then I say the Y calculations can be performed identically.

We may say it is three times Z which is the output capacitance, 3 because 1Z + 2Z + 3Z, so the next capacitance seen by Y is 3Z, so Cout for this gate is three times each so it is 3 Z-set into the gate which is again 2 input NAND so 4/3 is the G for that divided by the best effort stage effort which is 4. So 3Z divided into 4/3/4. Z is of course is 1.5 as we calculated. So Y also turns out to be 1.5.

So I can then therefore evaluate the actual capacitances seen by each stage even not knowing this, provided I know what is the output capacitance, what is the input capacitance, how many stages in which signal is propagating, what is the kind of gate we are driving and if I know this and how much are found out for each of them. If I know these numbers I will be able to tell what should be the capacitances here.

And once I know capacitances, I know capacitances are proportional to the width of the transistors and therefore I can evaluate the width and therefore I can say I know the best delay if I put these match values of W/L in respective transistors. This example essentially is what our tracks of everything what I said because given a network I should be able to see size the transistors such that the delay is minimum, that is the idea of all design of a circuit.

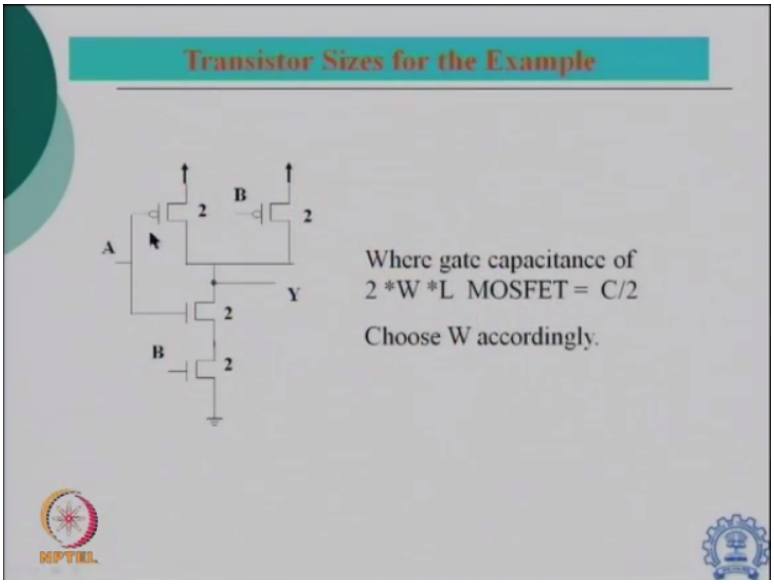
We could always do all these analyses by putting on a spice, putting some arbitrary values of Y and Z and can start simulating. However, one has to accept some value initially of Y as well as Z to actually start simulation. Now if your guesswork for those values is not correct or not close to what you are going to get like 1.5 of 1.5 Y and Z. Then what will happen that the simulator will never able to converge to these values or even if it converges.

It will take hell of a time to know what are the values of W/L corresponding to Y and Z. This kind of analysis we can see in a very short time back of the envelope kind of situation. I was able to find out what is the value of 1.5. Now I am not trying to say this takes care of every parasitic in real life so what you now do is since parasitic cannot be larger than the actual capacitance, so

we say once we know this value of XYZ or whatever in the line and the network.

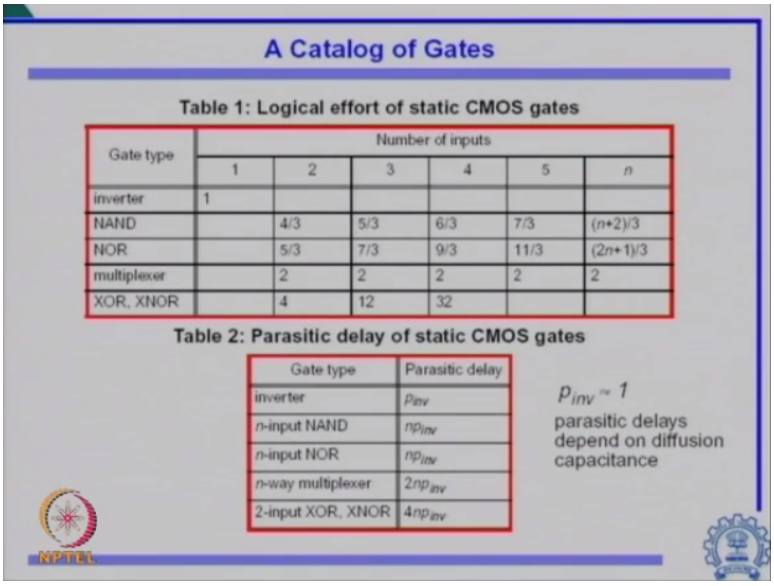
We can then probably put those values of W/L correspondingly everywhere and then start stimulating, the spice result will come, even for a very, very large circuits the result can come in a very short duration.

(Refer Slide Time: 23:30)



This is what exactly I said, once I know the typical transistor gate you take NAND gate for example, I know 2 into W/L is C/2 so corresponding to this width of those transistors can be chosen.

(Refer Slide Time: 23:45)



This is a table which essentially gives you a catalog of gates, for example the first part essentially says the logical effort of static CMOS gates for example you have an inverter NAND gate, NOR gate, multiplexer, XOR, XNOR and these are inputs, number of inputs is 1, 2, 3, 4, 5, 6 on N. For an inverter there is only one input no questions, for a NAND gate, the 2 input NAND gate has a logical effort by N + 2/3 so 2N 4/3 is a 2N + 1/3, it is 5/3.

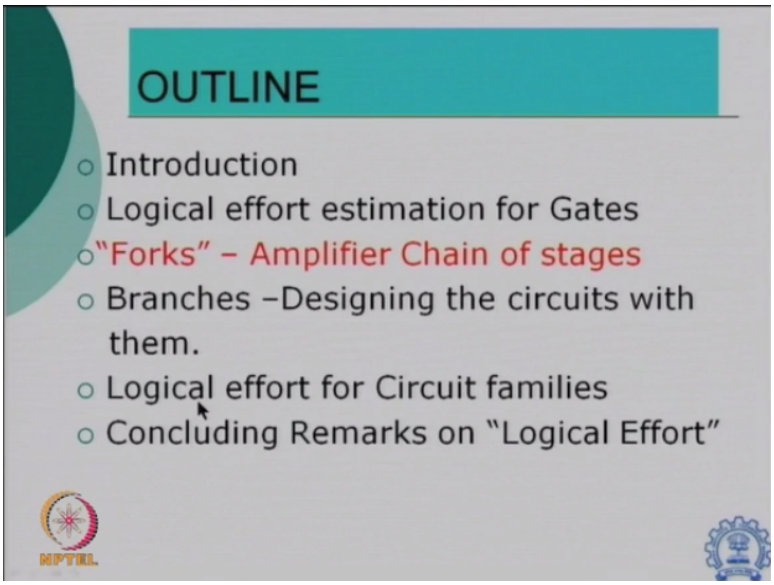
NOR gate it is 5/3, 3 + 2/3 is 5/3, N + 2 is the number, so 4 + 2/3, 6/3, 5 + 2, 7/3 and N + 2/3. By a similar argument we know NOR has a logical effort of 2N + 1/3 so one can see for 2 input it will be 5/3, for 3 it will be 7/3, for 4 it is 9/3 and so on. If you look at the multiplexer which I designed earlier one can see the logical effort for all of them.

Because there is an area of multiplexer in way we are doing individual stages only 2 input or 3 input, per this a same number of inputs is independent and therefore the multiplexer has typical logical effort of 3. Now XOR or XNOR by same thing, if you have 2 input XOR or XNOR you have a logical effort of 4, 3 input as 12, 4 inputs at 32 so it actually rises extremely fast as the input starts rising.

Corresponding to the theory which I just developed for the inverter, total static parasitic delay for static CMOS. This is a table gives you finally, the inverter has 1P inverter which is one unit and input NAND gate will be N times P inverter so N as the parasitic delay. For N input NOR it is NP inverter. For NV multiplexer it is 2NP inverters and 2 input XOR will be 4NP inverter. So once I know the parasitic delays of each of kind of gates I am going to use in a static CMOS.

I know the logical effort and if I know the electrical effort for a given network I should be able to actually design any network for its best performance.

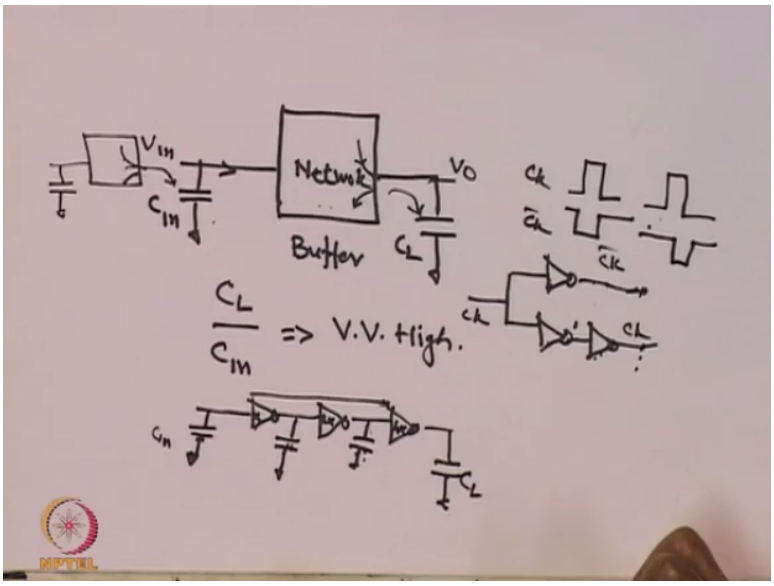
(Refer Slide Time: 26:12)



Having done so far a static CMOS and normal combinatory logic one of the interesting feature of a design of a circuit is essentially what we call a fork or amplifier channel stages. This is very, very important for simple reason that in a normal digital hardware you see clock is being distributed all across the chip. Now since the clock is a signal which is moving on an interconnect line, an interconnect line acts like more like a transmission line.

So there is a delay from the clock generating pad to the end of the chain where ever the clock is moving, so it may happen that clock may not remain clock because of delay, it may become clock bar at some stages and therefore every now and then we must actually have to amplify and bring it back to its original clock timing. If that has to happen we need so-called amplifiers.

(Refer Slide Time: 27:16)



There is another issue which occurs in normal life is the following: For example, you have a network and you are driving through some input some network, it has an input capacitance of some Cin and it is driving an output capacitance another this CL or maybe Cout, which is a typical circuit we have but the problem right now is that Cin/Cin is very, very high. Now this case occurs most times.

Because this is the output of a chip possibly, the output load may be very high which we are not even aware when we design and your input is decided by the stages which you have already put so this network in that case will be a buffer stage so this will act like a buffer then in which whose input capacitance is seen, output capacitance to be driven is very large CL. Now the question arises if I want to retain speed that means I should be able to charge the start the output node faster this is might be zero for example.

Then, the problem is since the capacitance is larger CLDV/DV0/DT which is the current required to charge this capacitance, will be very large because the load capacitance is very high which means this network must be able to provide or sink this large current. The larger current can only be supplied or sunk if the sizes of the output stage of this network is very, very large because W/Ls are very are very large only than it can provide you large currents.

If this buffer stage has very large W/L obviously your Cin also will increase. So when the signal

actually is coming it will take much longer time even to charge your input more, which essentially means everytime you want to improve this, there is no way you can increase the current without increasing the last stage input capacitance. Now it has to be driven by another block.

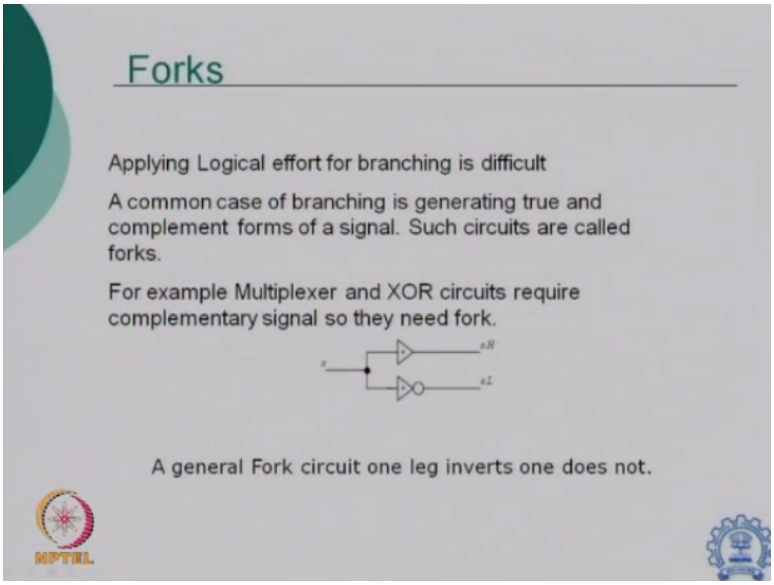
Since this is increased I need larger currents coming from the earlier stage, now if these are to be larger, I will get this capacitance large and there is no end to it. Now how do we get rid of this problem, so what logical effort method suggests that this buffer can be broken into stage of inverters. And the way it should happen that this capacitance to this capacitance ratio is so chosen that it does not increase drastically the W/L for this but does increase it.

And in stages, you can see this is X, this will be 2X, this will be 4X, at least double I am saying and then finally, maybe N stages, and finally you may have your CL. If you do stage kind of inverter in this then we can say the input to output, individually you do not have to drive too much because the ratio electrical effort will be not very large and since this is also now larger, here also it will be smaller.

And probably you can optimize the delay along this path as we did just now. Now the other problem which I am saying in the fork is the clock and at times even the clock bar which are required simultaneously in this circuit. If you see this is clock bar and this is clock. Now in this circuit also one can see there is an inverter delay of 1 here, there is inverter delay of 2 here, so this is essentially not just out of phase.

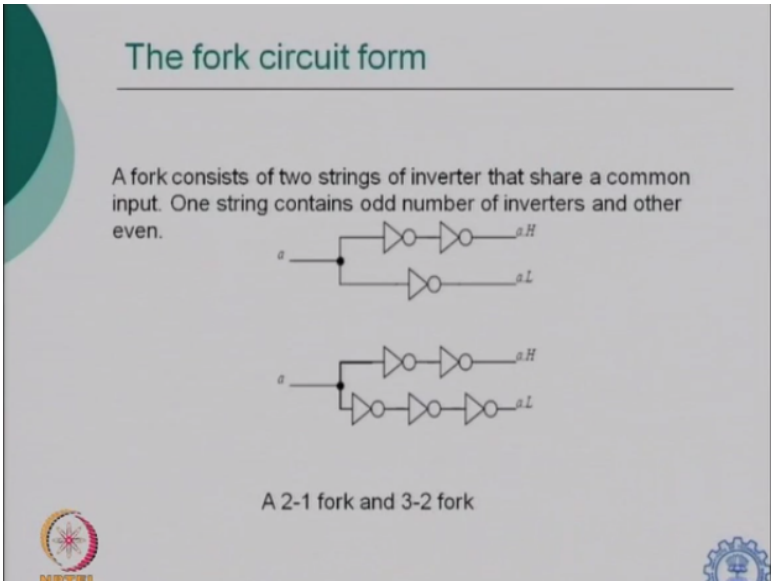
Because normally what I expect if this is my clock this is my CK, this should be CK bar but if there is an unequal delay, this circuit, this input clock may actually show you worst-case, this kind of this and they may actually never be clock, clock bars at a given time. Now this situation can be sorted out if somehow the delays of two inverter is made equal to delay of this inverter so when clock and clock bar appears they are actually out of phase as is desired, this is called fork.

(Refer Slide Time: 31:55)



So coming to fork, okay whatever I said I think it is already shown here. I do not have to repeat for example, for example multiplexer and XOR circuit require complementary signals so they need forks, a general fork circuit, one leg inverts, one does not is shown here as just now I discussed.

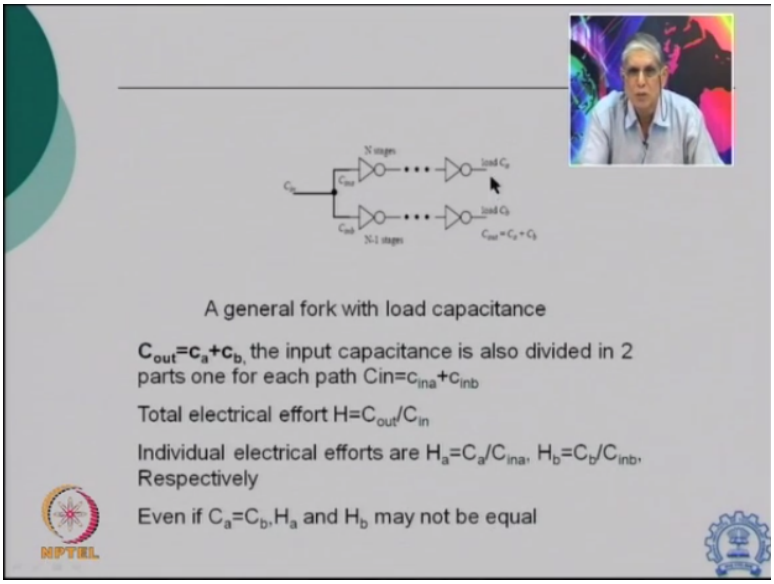
(Refer Slide Time: 32:15)



So there can be number of ways this fork can be created, one is you have on the one leg of the circuit, network has 2 inverter, the other is 1, which will allow therefore if you can equate the delay on both sides then I will have clock, clock bar as I will need. Alternatively, I may have a combination of two inverters on one leg and three depends on the load at the outputs I am going to drive, okay.

Now, this is called two one fork, three two fork, it can be three four fork and so how much kind of this numbers we chosen so the delay is minimized, is the aim of this part of the lecture which I am now talking.

(Refer Slide Time: 32:54)



A general forgave therefore shown here, you have load capacitance and then output capacitance on both sides may need not be same as well because this is very relevant because clock may be driving some circuit, clock bar may be driving some other part of the circuit and their loads may not be same their input capacitance therefore load to these may not be same. So let us say Ca and Cc are the loads of the upper.

And this has N inverter stages, this is N -1 stages, for example shown here it has an output capacitance of Cb. Both have a net input capacitance of Cin and we say now we want to know for making delays equal long, N and N1, N + 1 stage and N in -1 stages, we believe that if I can size these chain of inverters on this side and chain of inverters on this side and chain of inverters on this side such that the path delay along this line is same as path delay along this side.

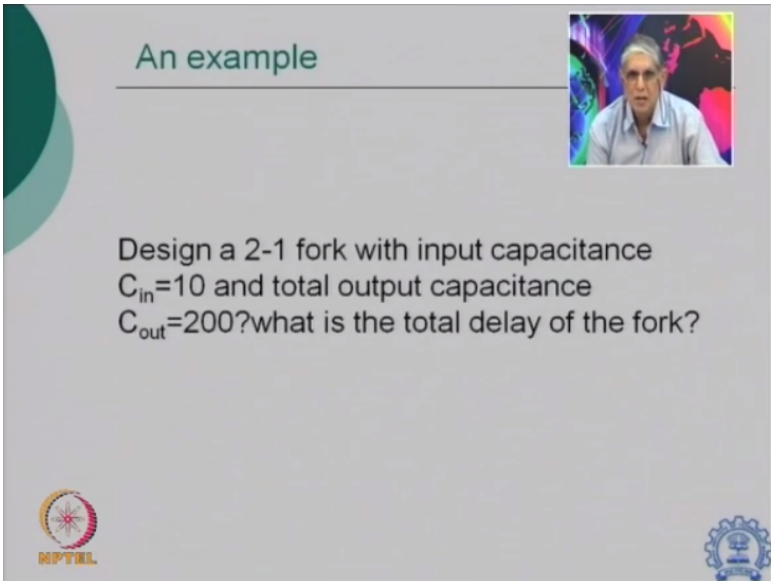
Then we may say clock, clock bar will exist even if the output load capacitance is not equal. However, the net output capacitance is essentially Ca + Cb, the input capacitance is also divided into two parts seen A, C and B okay and the total electrical effort from input side to the output is

essentially w call Cout/Cin which we can therefore calculate.

Individual electrical efforts will be actually for example this one will be Ca/Cina and for this case Cb/Cinb, so I know individually I can calculate electrical effort for this blog, electrical effort for this blog and even if let us say Ca, Cb, I mean best case possibly, Ha and Hb may still not be equal. This is very relevant thinking because when I am equating the delay the capacitances which I will get from this side and capacitance.

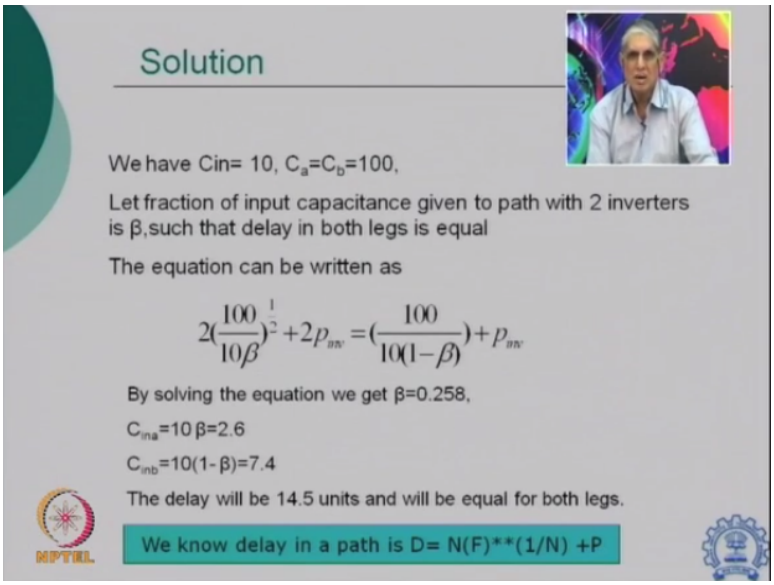
Because the other stages are not same will not be same. Therefore, even if the Ca is equal to Cb that is Ca is equal to Cb, H1a may not be same as Hb that is Cina will not be equal to Cb, and therefore the evaluation should not take care that it is always Ca equal to Cb because even if it so we have to still calculate Cina and Cinb.

(Refer Slide Time: 35:22)



Here is an example, we have to design a 2-1 fork with input capacitance of 10 units and total output capacitance is 200 so what is the total delay in the fork, okay.

(Refer Slide Time: 35:36)



Now, we say Cin, right now we assume Cin is 10 and let us say for the simplicity case, which is Ca equal to 200, so we give 50-50 to each of them, 50% so Ca and Cb is 100. Now we already said the two paths will have Cina and Cinb as the values for their input capacitance, and we know Cina is equal to Cinb which is equal to the net Cin, which is 10. So we say that let fraction of input capacitance given to part two inverter is beta of that and the 1- beta is therefore given to the upper part or other part.

And then we calculate the path delay along the two paths. For example, let us look at the circuit, the inverter, so we say 100 is our H, G is our two inverter, so it is this. Ten times beta is essentially the ratio of this capacitance, beta times this. There are two stages, let us say two forks, so 1/2, so 2 is because of that forks, so 2 into 100/10 beta to the power half + 2 inverters, please remember this is the path delay for the 2 inverter site.

This is the path delay for one inverter site and therefore and what we are saying if we want the circuit to perform as clock, clock bar, the delay should be equal. So we say two fork system will have 2 into 100/10 beta to the power half + 2p inverter must be equal to 100 divided by 10 into 1- beta now because there is only part of it capacitance is given to 1 inverter. So 10 into 1- beta + 1 inverter, therefore one inverter.

Now we equate them. Since I know this, this is a simple equation, one can see. Since I know P

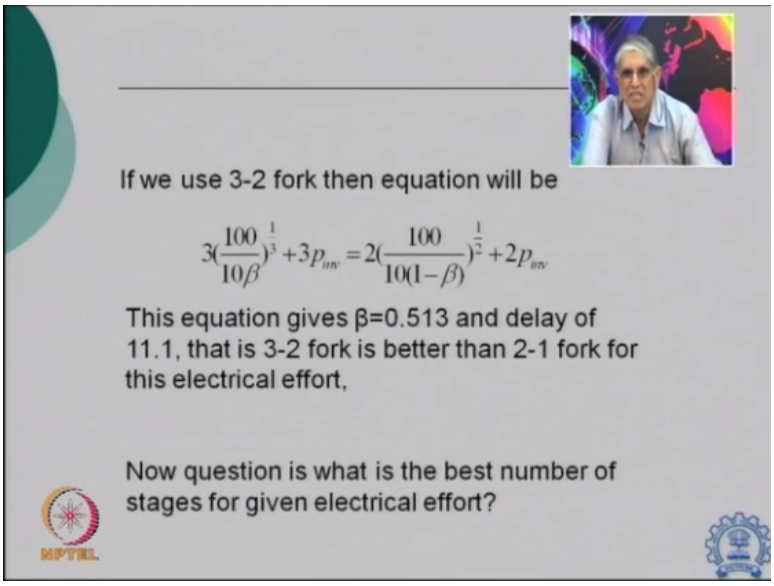
inverter, I know every other thing except beta and if I solve this simple equation, then I get beta is equal to 0.258. Since I have divided my input capacitance in a fraction of beta, so Cin 8 is 10 times the beta, which is two fork system, which is 10 into 2.58, which is roughly 10, 2.6 and correspondingly the one fork capacitance will be 10 times, 1- beta which is 1 - 0.258 is essentially 267.4 into 0.74 into 10 into 7.4.

And if I substituting either of them, they are equal obviously, one can by using this formula, one knows the delay along either of the path is please remember you calculate this because you do not have to do this 100 upon 10 - 1 - 0.258, 0.26 + 1, if evaluate this number appears to be 14.5 units subtly and since they are equal both are of a 2-1 fork has equal delay of 14.5 units.

The input capacitance to the two fork one should have a BC and AS 2.6 whereas Cinb should be 7.4 units, such that the net capacitance from the driver stage is 2.6 + 7.4, which is 10 units as provided. So one can now that by actually making a choice of 2 or 1 or 3 and 2 but in case of 3 and 2, this will become 3 into 100/10 - 1 beta to the power 1/3 + 3 be inverters and then we equate and re-evaluate beta for 3-4 stage, any number of such stages can be good enough.

And we may evaluate then for 3-2 stage, again the delay. If we figure out that the 3-4 gives lower part delay than 2-1 fork than we go for 3-2 park. If we do also analysis for 4-3 fork and re-evaluate the delay if that delay does not change very much from the 3-2 fork then we need not go for 4-3 fork, but we continue with 3-2. If that does not happen, we may try even 5-4 fork and keep doing until the delay is minimized.

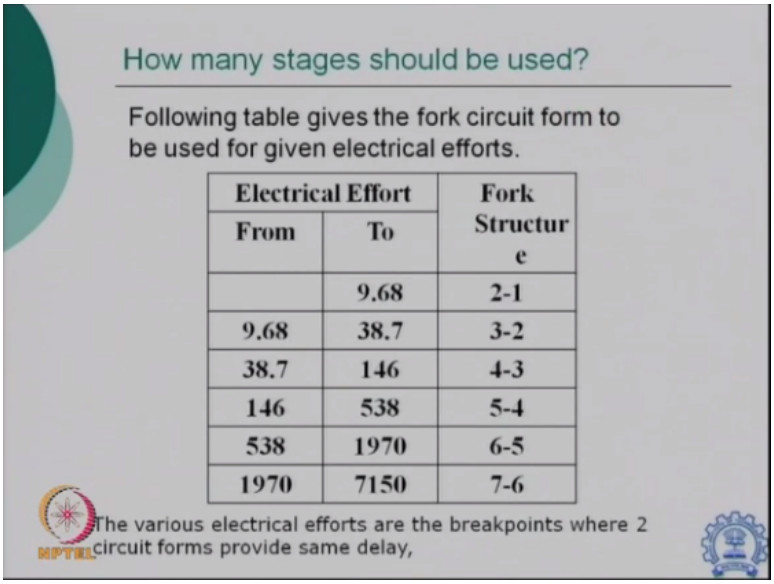
(Refer Slide Time: 40:03)



Just now I said, I do this calculation for 3-2 and one can see this equation gives beta 0.51, delay of 11.1, so obviously since the delay there were 14.5 here it is 3-2, it is 11.1 A 3-2 fork is always better than 2-1 fork for this electrical effort. Please remember for different electrical efforts, this value may be different and may have not same, if both Cout A and Cout B are not equal, these values will be different for 3-2 and 2-1.

It may be found that 3-2 may not be as good as 2-1 or 2-1 will be as much as sufficient to work with. Now question is what is the best number of stages for a given electrical effort.

(Refer Slide Time: 40:48)



Electrical Effort		Fork Structure
From	To	
	9.68	2-1
9.68	38.7	3-2
38.7	146	4-3
146	538	5-4
538	1970	6-5
1970	7150	7-6

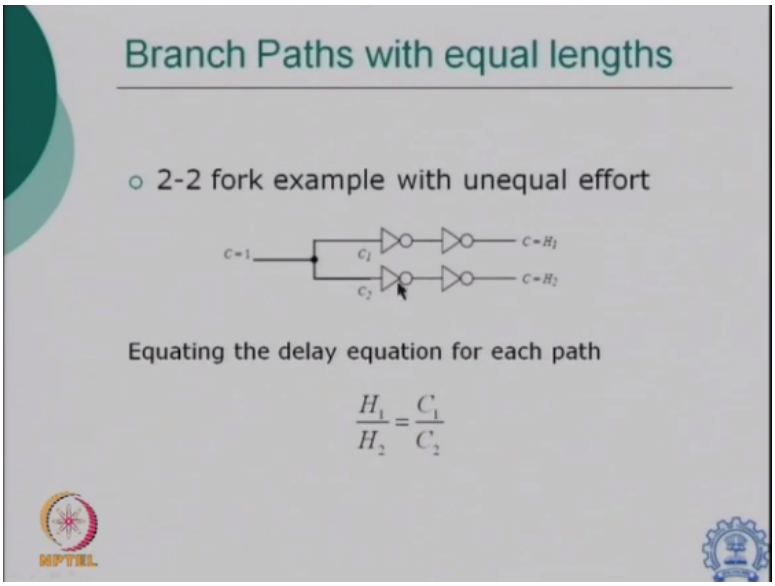
So here is a table which doing this analysis which I said for every such stage people have already

done this, this Sproull and Sutherland book already gives this number. So, for example if your electrical effort is from say 1-2 or 0-9.68, it is found that 2-1 fork is good enough for the minimum delays you can get. 3-2 you do it but it will not improve very much in delay. If your electrical effort is 9.68 to 38.7.

Then we say 3-2 fork should be used and by similar logic if it is 1970 into 7 that is very huge loads if you are looking at the output, then you may require a 7-6 kind of fork requirement and depending on break points you are looking for, you must be able to decide which one of them should be used. So, in case of buffers at times the load is very heavy at the external side, then the buffer stage should have more than 3 or 4 clock stages to put the dive out.

So, is that point clear that what is the advantage of fork, because forks tells us that we can equate the delays but we can figure out how many stages should be used in forks in upper and lower arm such that the delay is minimum and equal to on both arms. This interesting feature, you must see that at this point, let us say from here to here, they are same values, so whether you use to 2-1 or 3-2 will always give you same daily and that you can verify, that is how the break-out points are actually obtained.

(Refer Slide Time: 42:23)



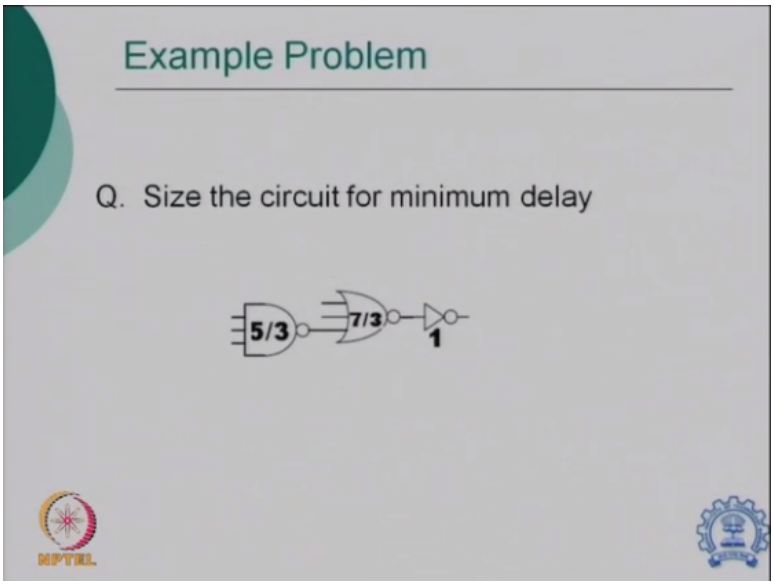
There is another interesting issue which is shown here, if you have an input capacitance driving in such two paths, unequal effort branch 2-2 fork example with unequal efforts. H1 and H2 are

unequal efforts. Equating the delay equation for each path, since they are same identical this. So, H1/H2 is C1/C2. C2 is here, H1 is equal to C/C1. C/C2 ratio is essentially H1/H2 is C1/C2. So, since I know for one I can evaluate, the other can easily be evaluated and figure out what should be C1 and C2.

Now, if you have unequal effort that is the number or kinds of gate on one path is different from the other, even then you need not worry, F1/C1 will be equal to F2/C2, where F of course you can figure out by the simple path effort GH way. Once I know, if there is a branching in the leg, we can still evaluate B, F leg / F others F leg, we can then find the path effort F for both paths. Stage effort we know by F to the power 1/N.

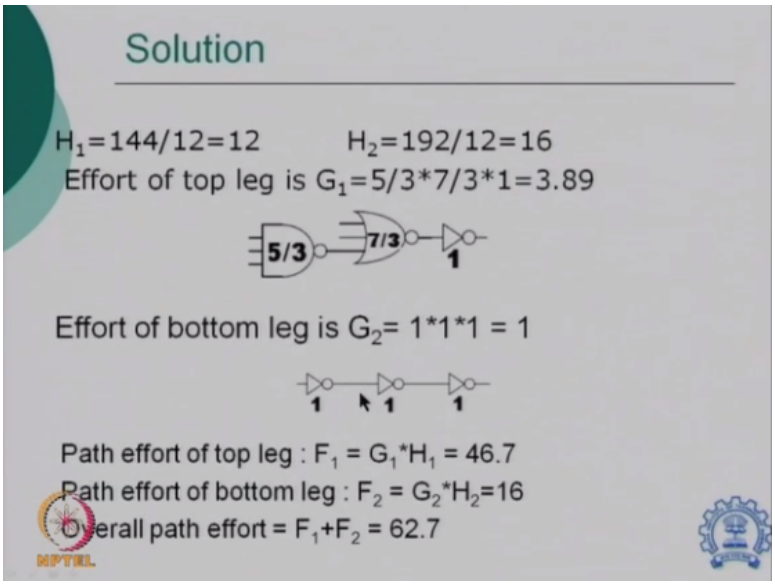
And since we know F/C1 and F/C2 for both F are known and C1-C2 one of them will be known. I will be able to get C1 or C2 and using F we can go back continuously back until the first inverter is obtained. So, one can do 2-4 unequal effort or unequal electrical effort or unequal path effort and can still do the analysis.

(Refer Slide Time: 43:58)



Here is an example for the minimum delay. You have a NAND gate which is 3 input, you have a NOR gate which is 3 input and is driving an inverter at the output. The G for this is 5/3, G for this 7/3. Please remember this is N + 2/3. This is 2N + 1/3. N is 3 input, so 2 into 3 is 6 + 1/3. This is N + 2/3, that is 3 + 2/3, that is 5/3. Inverter has a logical effort of 1.

(Refer Slide Time: 44:29)

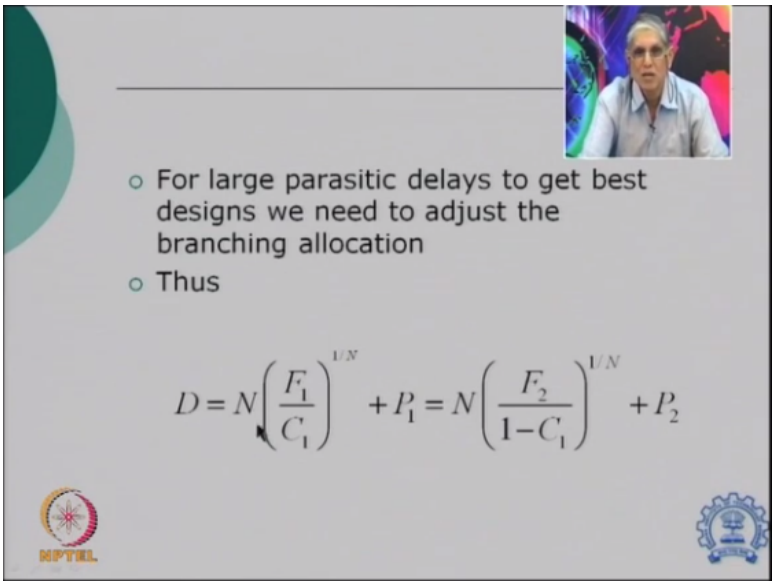


Let us say each of them output has H1 is 144/12, H2 is separate capacitance driving, H2 is 192/16. The effort of the top leg is 5/3 x 7/3 is 3.89. Effort of the bottom leg is only inverters, for example in 1 x 1 x 1 x 1 is 1. This is one path, this is another path and one can calculate F1 GHB. B of course is not existing here. So, only GH. Path for lower leg from here is G2, H2 16. Overall path effort is the sum of the two F1 + F2.

Now, if you want to size, this is the kind of circuit I am saying. B is equal to F1 + F2/F1 which is 13.4. F is G1B, H1 is 62.7. There are three stages 3.97 working backward I can evaluate as we did earlier, all capacitances. Similarly, one can do back calculation from here to here by calculating the minimum stage delay and then going backwards to evaluate.

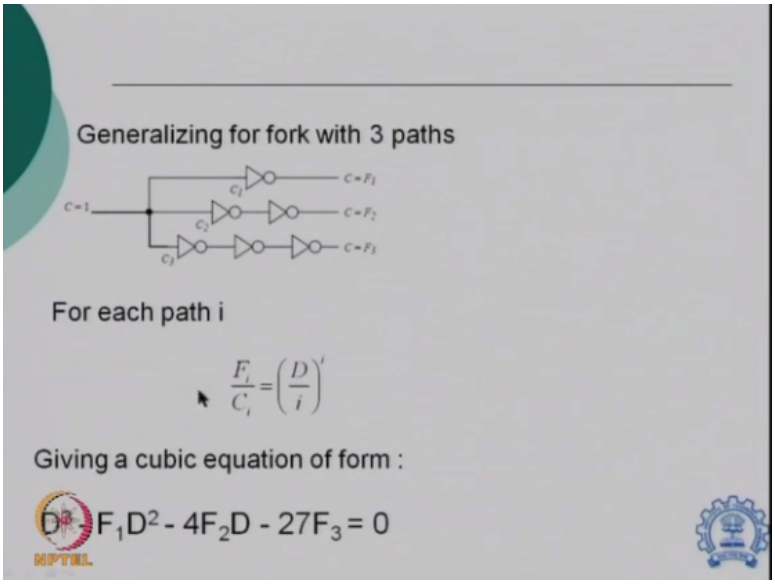
Please remember, when you do the last calculation, this capacitance accuracy will be decided by when you calculate Cin for this using the stage effort which is common to all then it must get the value of 3. If it does not please remember your analysis of calculations have gone wrong because this is fixed to start with Cin. This is fixed start with Cout. So, when I am going back calculations, from 12 to input this must appear 3 at the end of the day. If it does not, your earlier calculations are also wrong, please go back and verify.

(Refer Slide Time: 46:22)



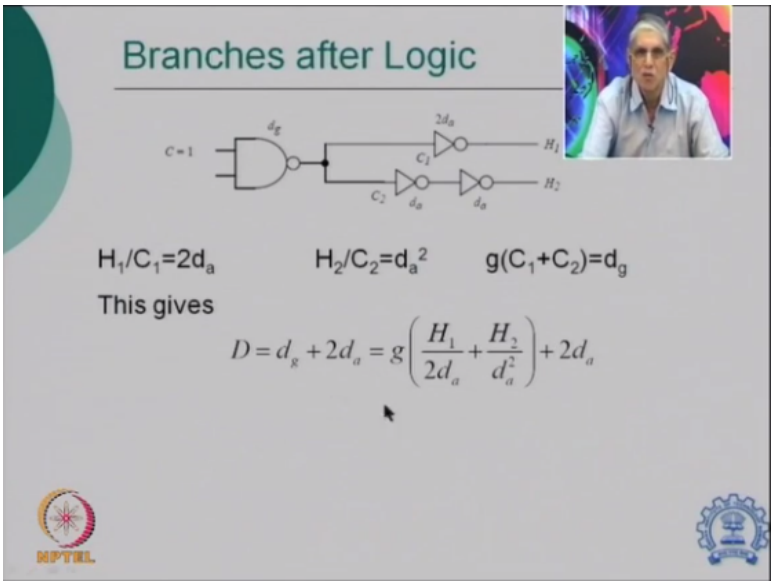
For large parasitic delays to get best designs, we need to adjust branching allocations. So, we can see this must, that is what I said, this is the nutshell what I said, N x F1/C x 1 + P1 = N x F2/1-C1 to the power P2 and if I can do this I can minimize the delay for any of the branch efforts.

(Refer Slide Time: 46:48)



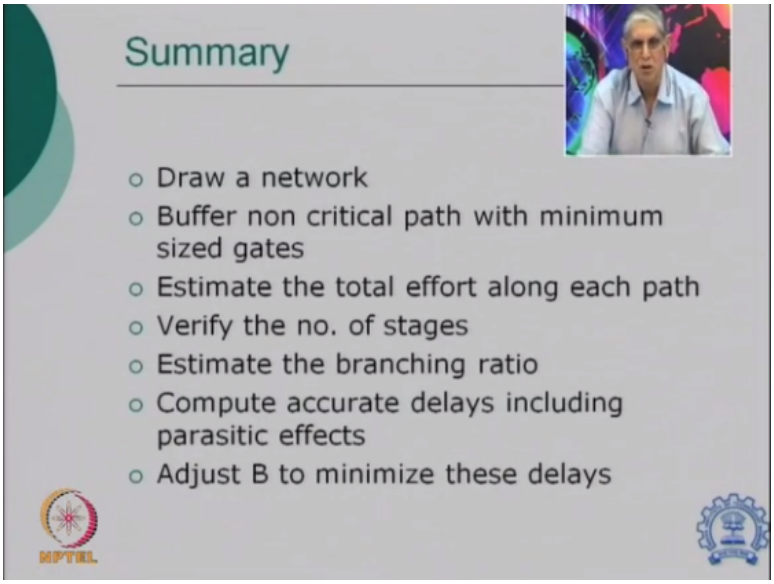
We can have more than one branch, you may have 3 to 1 with different F1, F2, and F3 here and by same method which I say, instead of now 2 it will be 3 and therefore you will have some kind of cubic equation in this case for each of this when you equate.

(Refer Slide Time: 47:06)



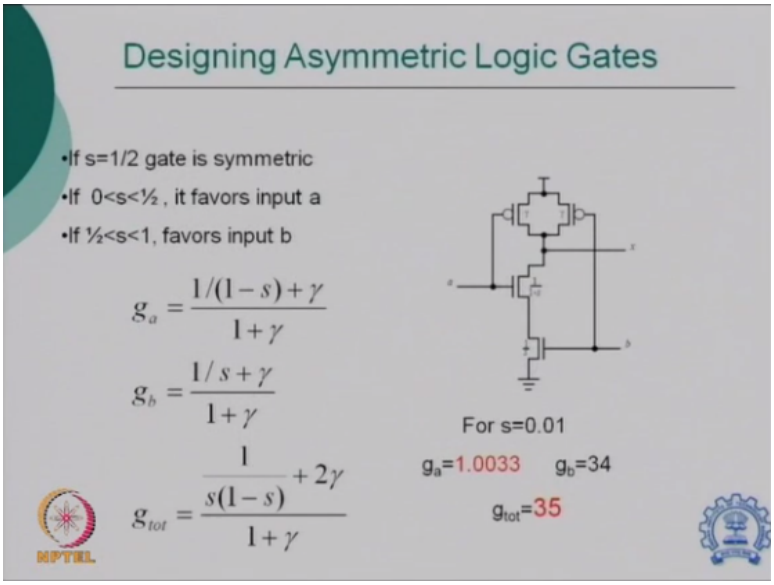
And if you solve this, one can always still optimize the path delays.

(Refer Slide Time: 47:16)



You can read the problem solved in here because I must finish on my talk fully before today. So, a typical summary for this is draw a network, buffer non-critical path with minimum size gate, estimate the total effort along each path, verify the number of stages on that path, estimate the branching ratio, compute accurate delays including parasitics, and adjust the path B to minimize these delays.

(Refer Slide Time: 47:42)



Okay, so this is essentially what I say, given a branch or fork system, any number of branching can be done, delays can be minimized and this is actually what a normal circuit will have. Please remember that each circuit, we may draw more than one blocks and therefore the essentially branches will be constantly available in hardware. So, analysis of delay all along the path is a very crucial way.

And it can be done using logical effort much simpler way and much faster way and can also give you minimum delay for which capacitances can be calculated. The another part of this logical effort since we have been taking a normal, you know, simple symmetric logic in which the template inverter was 2-1 and then we always say make it equal to 1 and make it equal to 2, there is another way of doing.

Many circuits may not have such symmetries and we may have to do asymmetric logic in your life. For example, the point I am saying if S is half, this is 2 and this is 2, that will make one and this is SNS, so it essentially means that this is 2 and this is 2, so it is 2 essentially so one can see that if S1 is half perfectly fine, there is no problem in actually getting 2:1 ratio on this, but if F is less than half.

Essentially you say A input is favored because S is half mean this value will be larger and less than half means this value will be smaller, that means A input will be essentially favored. If S is

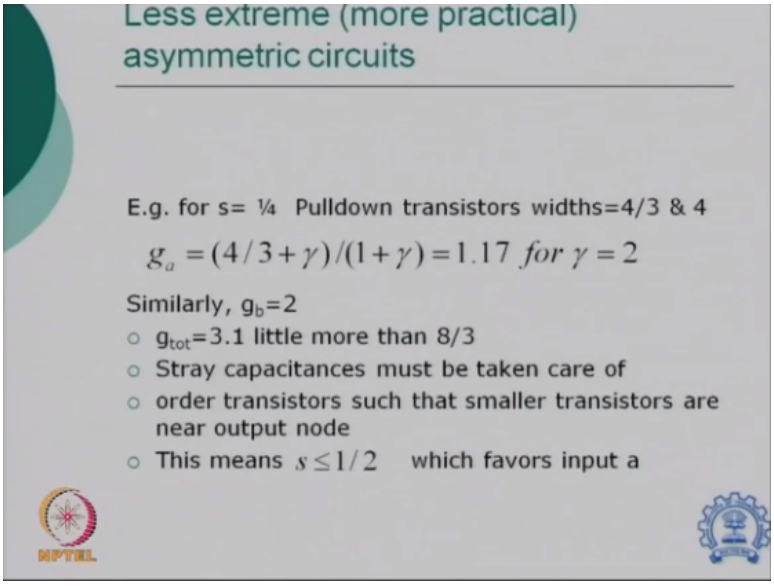
greater than 1, the B input will be favored. So, this path will have more current to worry about, limiting point. In other case, this will be essentially decided. Since they are not symmetric, one must evaluate the logical effort for both paths.

And then also get the delay for each of this case. For example, this is since gamma time. So, A is 1/1 – S + gamma upon 1 + gamma. GB for this size is 1/S + gamma upon 1 + S and the total of the 2G GA + GB is 1/S1. Please remember if I substitute S is equal to half, this is half 1/2, this is 1/2, 2 x 2 is 1. So, 2 + 2 gamma/1 gamma. Okay, sorry. This is half, so this is 2. This is 1/2 means 4.

So 4 + 2 gamma / 1 + gamma which is 2 + 1 that is 3 which is what essentially you get in the case of a NAND gate of one size. Now if you see, you can see this is 4/3. This is 4/ if I take gamma is equal to 2, then it will be 4/3. So, this 4/3 system which is for a normal NAND gate will now change to a total of 3.5 and individually GA has a logical effort of 1, whereas GB has a logical effort for 3.4, if S is very, very small.

Please remember, if A will be favored if S is less than 0.5, B will be favored if S greater than 0.5 and this is the value which actually tells you. So, continue working with new logical effort, if you are using asymmetric gates. All that you now need is very important issue has been created. The discharge and charging transition should be different in the two cases and therefore one has to actually derive TPHL path separately from TPLH paths and then evaluate the net delay.

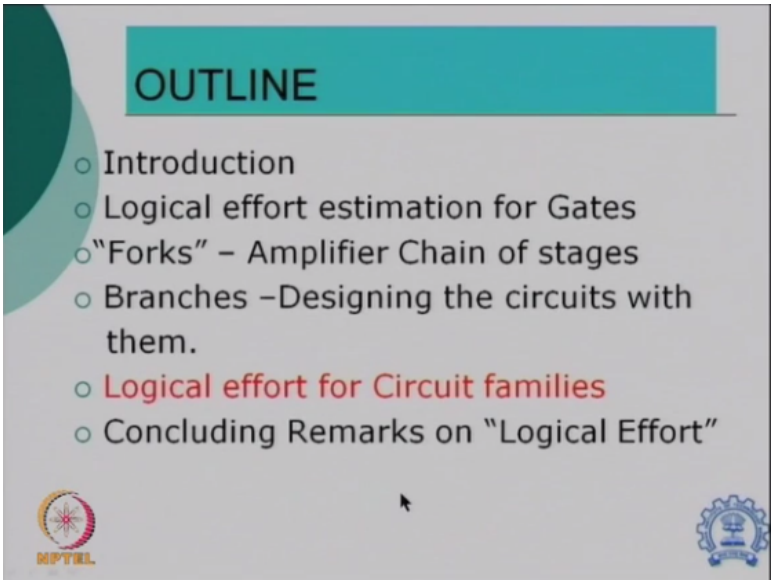
(Refer Slide Time: 51:25)



Slightly, not very simple as trivial has had made but can be done, still analytically can be done. For S is equal to 1/4 pull down transistor width will be 4/3 and 4 and therefore GA will be 1.17 for gamma of 2. Similarly, GB will be 2. G total is 3.1 little more than 8 x 3 which you would have got otherwise. The stray capacitances must be taken care order of transistor such that smaller transistor are near.

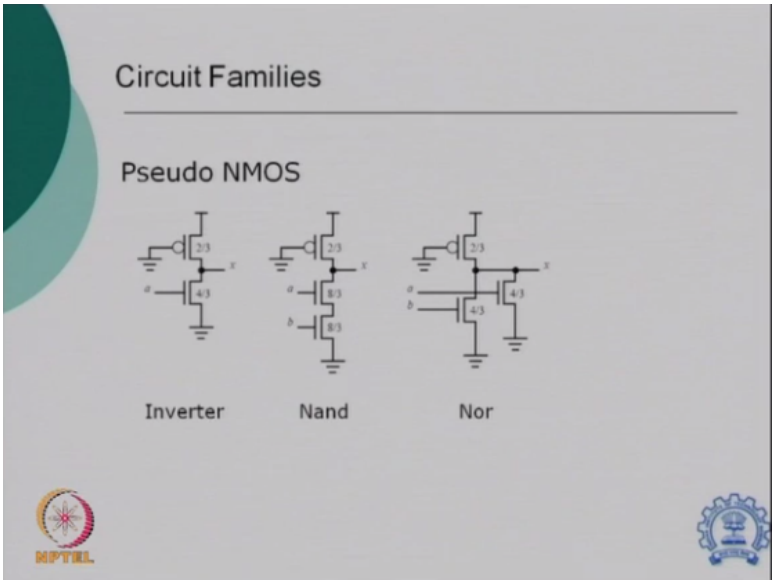
If we know a normal design that the transistors which are smaller in W/L should be kept closer to the output note, so change this A or B correspondingly and this means S is less than half which favors input A.

(Refer Slide Time: 52:09)



If not the last but one part of this course, we have looked into normal static CMOS. We also look for branching effort. We also look for forks. Now we look for other circuit families. One of the famous circuit family in CMOS is called pseudo NMOS.

(Refer Slide Time: 52:33)



This is essentially chosen in particularly when you want saturated kind of loads to be put to an N Channel device. This is like saturated load transistor inverters but this is not necessarily saturated but depending on the state of this, this P Channel transistor even if it is gate is grounded may actually move from saturated to unsaturated. So, it is a varying in resistance here. This is of course fixed by us by 4/3.

The ratio is still kept 2-1 4/3 and 2/3 is for pseudo NMOS. There is a NAND gate using pseudo NMOS. What is advantage of pseudo NMOS. The advantage of pseudo NMOS can be seen from here. In a static CMOS, I would have required two P Channels. In the pseudo NMOS, I require only one. In the case of NOR, I have 3 transistor NOR. Whereas in the case of static, I would have put 4.

The other things are kept same. This each is to be 4/3, so it is 8/3, 8/3, double around is 2/3 whereas this is kept same 2/3. Please remember, now it is slightly different. Since, it is a load inverter kind normal, we know the load size should be smaller than the driver transistor. Please remember in the case if this is the register, this register value should be higher so that the viewer

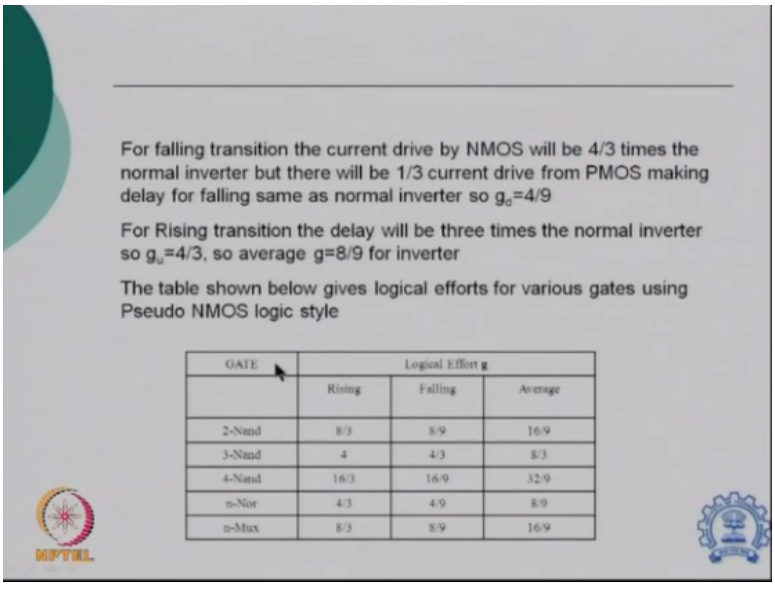
low is lower than the minimum required for less than VT.

Now, to make this typically, we use this transistor half of size of this and that is the number which is. Please remember it is not like a template transistor of a static CMOS where this was 1 unit, this was 2 units. Here it is the other way, this is kept as 4 x 3 whereas P Channels are kept 2/3, double the N Channel size, half the N Channel size is used in P Channel, so the resistance of P Channel is higher than N channels and because of that the (()) (54:34).

By same argument since this is to be 4 x 3, this is now new template for us. In a pseudo NMOS, the template is 4/3 and 2/3. If I convert it is NAND, it is 8/3, 8/3 series makes it 4/3. P Channel is single, so 2/3. In a NOR gate, your need either path to be 4/3, so each N Channel we have parallel need to have only 4/3, do not need 8/3 because they are not in series. Now, once I actually decide this, for each input one can evaluate the G 8/3 + 2/3 which is 10/3 divided by 4/3 + 2/3 which is 2, so divide by 2.

So, 8/3 + 2/3 is 10/3 divided by 2 is 5/3 per input is for the NAND gate.

(Refer Slide Time: 55:30)



By similar logic, I already said if P Channel and N Channel are not same resistance values because to keep view lower so that when there is a noise margin, the rise and falling transitions will have different path efforts or logical efforts. Please remember this is very important because

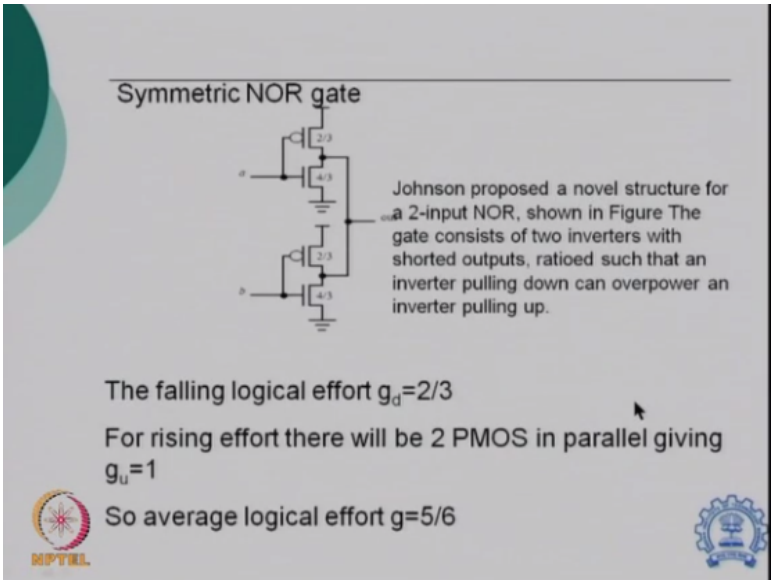
they are not equal. Charging transitions occur because of the P Channel transistor, discharge transition occur because of the N Channel chain.

And therefore the logical effort for rise and fall transition is different. For example, the table has given you all of it and remember load is always pseudo NMOS in our case P Channel device which has a value of 2/3. So, 2 input NAND has a rising transition 8/3, I just calculated, whereas this you can see why it is by 3 and the falling transition will be 8/9. Average of course if you take 8/3 + 8/9/2, it is 16/9. 3 input NAND, rising has 4, falling has 4/3, average is 8/3 and keep doing.

For example, N wave multiplexer. One can see from here, it is 8/3 for rise, 8/9 for fall, and 16/9. For a NOR, you have 4/3 for rise, 4/9 for fall, and average is 8/9. Once I know my logical effort for these kind of gates the rest is identical because all that is different from the earlier one was G is not like a static CMOS. One has now new G is available for these kinds of pseudo NMOS kind of gates you are using.

Since I know my logical effort for these gates, again the GBH will give me the stage effort and GBH to the power 1/N and + P will also give me the path delay. So, there is no difference once I get the G values which is different from static CMOS.

(Refer Slide Time: 57:40)



This is a symmetric NOR gate. We can see from here, these are two. This is very interesting

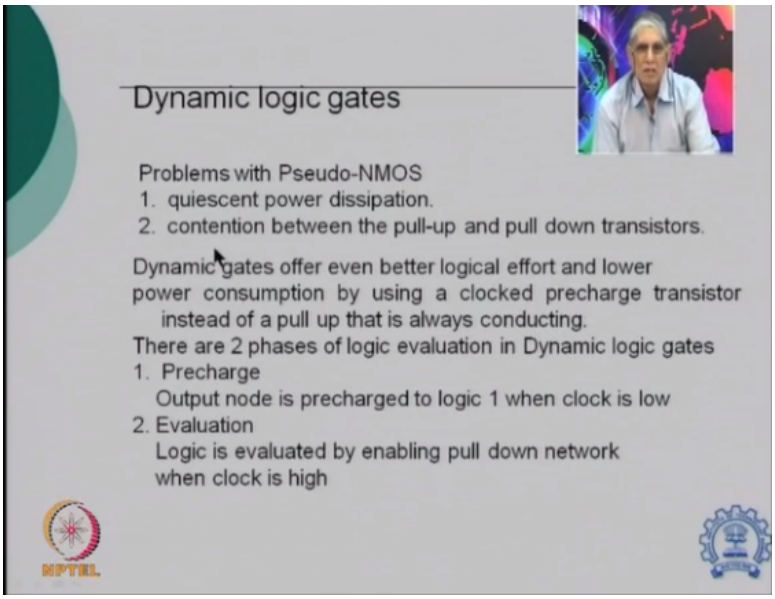
NOR gate. This is one inverter, this is another inverter driving this and one can then figure out if their outputs are connected, does this act like a instead of a pseudo NMOS, if I put a NOR gate in this fashion as shown here. Please remember, otherwise what is static CMOS was that, I put a P Channel transistor in series of this, two P Channel series and two N Channel in parallel.

Here is using asymmetric theory that is the gates are used 2/3, 4/3 and when I connect the output they are still acting like a NOR gate and then if I evaluate the falling and rising effort, GD and GU then I find for the rise transition since they are in parallel, G is 1 and GD is 2/3. So, the average logical effort is 5/6 for this kind of gate which is smaller than a NOR gate in the case of static CMOS.

So, if you are using a asymmetric method of doing symmetric NOR gates paralleling like this which is essentially proposed by Johnson way back is a novel structure and it shows that you can improve on the logical effort and if you know logical effort has decreased, we know GHB which is the path will also decrease and if your path effort decreases the stage effort decreases and if stage effort decreases, the value of capacitance will be smaller.

And also you will have a lower path delays. So normally we always believe that a NOR gate has to be static or something, so here is a static gate used as a symmetric gate and using this method, one can see one can actually reduce the path delays and therefore one can see that one speed up is possible with little difference. Please remember, in static also I would have required 4, in this also I required 4 but now it is faster than our normal static CMOS.

(Refer Slide Time: 59:57)

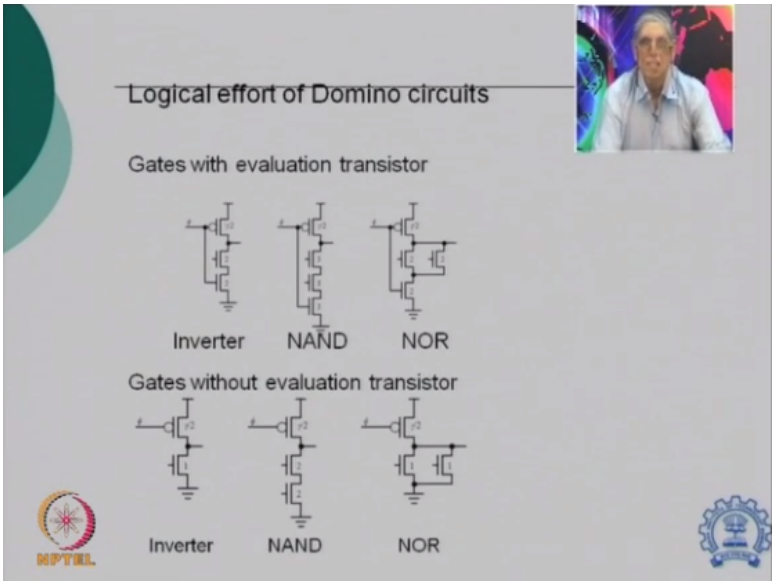


The next problem other than any such dynamic systems I do, since the P Channel device is always on in the case of pseudo NMOS, it constantly dissipates the power when N Channel is on, except for VGS less than VT which is only one transition time, small time in transition. Both N Channel and P Channel conduct currents and therefore constant power dissipations. If you look at normal dynamic gates.

One can see that in a normal dynamic gate which I will chose from latter, which is not pseudo kind. Only N Channel is acting at one time or only P Channel is acting one time and therefore net power will be minimized in the dynamic gate and here is some kind of, the only difficulty there is you need some kind of a pre-charge and evaluation phases. And then you have some kind of problem in all dynamic gate of charge sharing or logic not properly getting or the next logic changing earlier than.

All these of course can be solved by modified dynamic gates called Modified Domino or Domino NORA Zipper and once you know one can, then evaluate a G for them, then I will be able to again calculate the delays.

(Refer Slide Time: 01:01:07)



Here is for a Domino circuits, this is a gate with evaluation transistors. P Channel, this is an inverter, this stage essentially is the evaluation phase 5. This is our input transistor. When, there is goal 0, S goes 1, this is 5, so when S is equal to 0, ground is removed and charging occurs, pre-charge as recalled and S becomes 1. Whatever is the pre-charged phase here, depending on input here if it was 0, then 1 will remain.

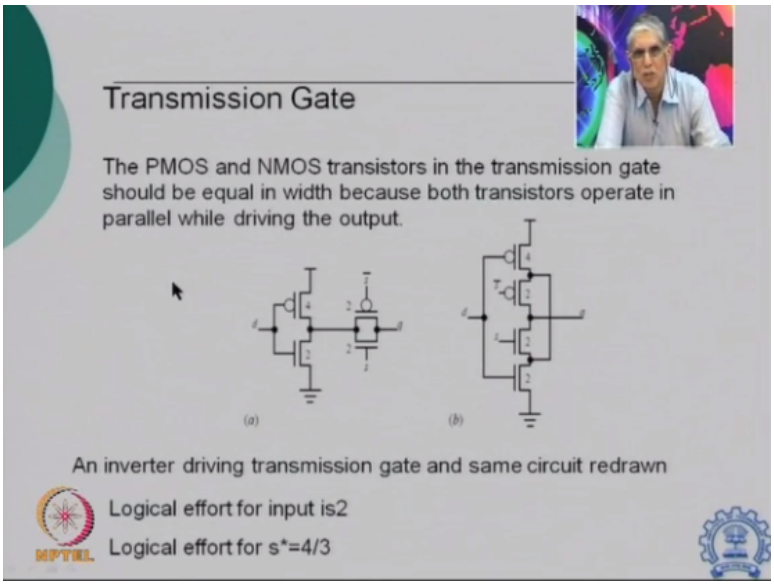
If it was 1 then it will discharge to this path and make it 0, and since when this was charging this was off, so no current in the power supply to ground on this side. When this was discharging and this was off, there was no additional current coming from P Channel. By same logic, I put two N Channel transistors in series here to make and I can correctly size them for series combinations and parallel combinations and one can then.

Please remember, the evaluation is on this side and charging is on this side. So for charging and for evaluation transistor, there may be a possibility of only one of them. Please remember, this is another zipper circuit in which lower N Channel is removed now, only clock is done through here. So, we say when this is 0, now assume at that time that this input, though it is available, this takes care of this current as well and still charges.

This increases little bit of power but it reduces one additional N Channel transistor and in that case probably the power may be slightly larger but at least you have lower number of transistor,

hence the area is minimized.

(Refer Slide Time: 01:03:06)

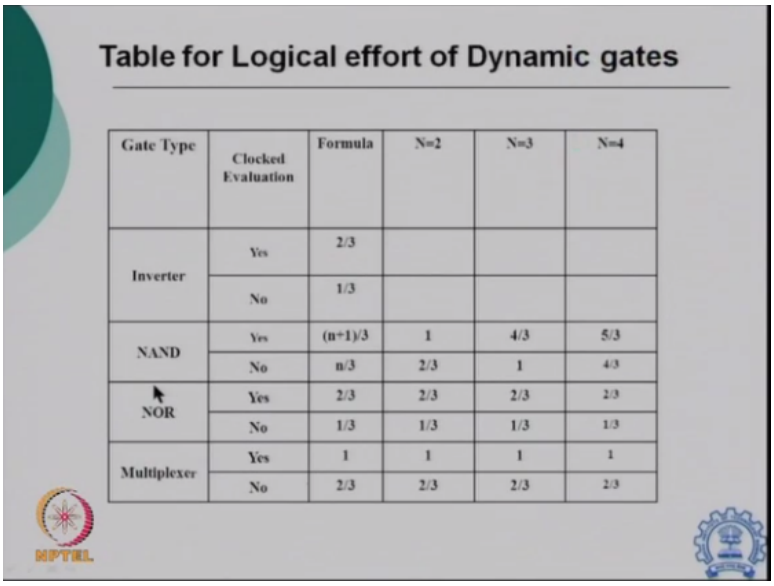


By similar calculations and similar thinking, we can do a logical effort for transmission gates. Here is an inverter which is passing through a pass gate, CMOS basket transmission gate. This of course has a ratio of 2 to 4 graduated, is usually 1 to 2 and this is taken larger, you know, it has to drive this capacitance CGD of these and therefore and finally the load so one normally puts little larger W/L here, this is my data, this is my transmission gate.

Whenever, this is 0, S bar is 0 or S is 1, this is on and the data is transmitted. When opposite occurs, then this S bar is 1 and S is 0 transmitting it off and data is retained. The same kind of circuit can be re-implemented on. There also are four transistors here, there are also four transistors, identical to this can be reconfigured and you can see here this is the data, the external this.

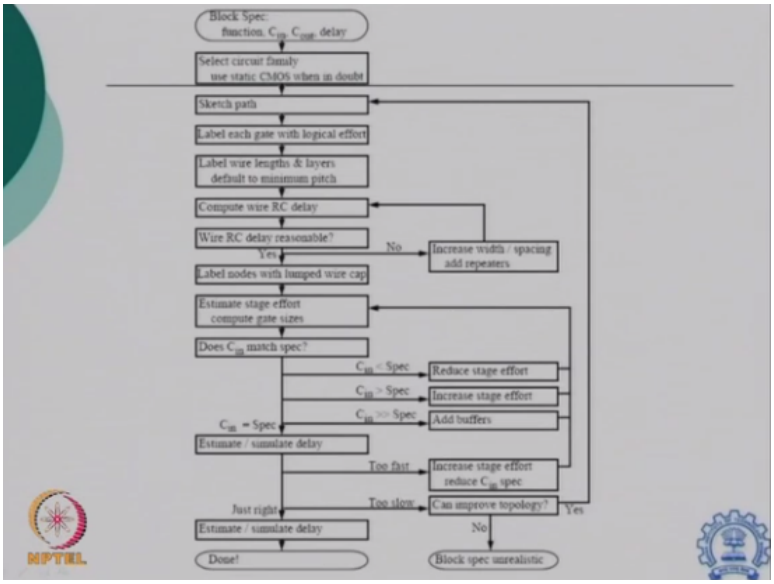
This is what the base control is there for the multiplex. If S bar is 0, S is 1, so this inverter data is then transmitted out corresponding to invert the bar and when S bar is 1, this is of S is 0, this is off, so no data can be transferred in the output. Looking at the numbers given to you for each input, the logical effort is just two, 4 + 2/3 that is 2. Logical effort for S which is straight line is 2 + 2/3, so 4/3. The net logical effort is 2 + 4/3 which is 10/3.

(Refer Slide Time: 01:04:53)



Now by similar argument, this table has provided logical effort of all kinds of dynamic gates, inverter, NAND, NOR, multiplexers, the method which I have just now shown is used to evaluate for 2-input, 3-input, 4-input, all of them are clock evaluations. If there is no clock as I just now said, then this becomes closer to static and therefore the logical effort is small.

(Refer Slide Time: 01:05:23)



So, typically before we quit this area, you can see from here this is a typical flowchart which shows how to do logical effort method to apply for your designs. Here is the block specification, essentially I need to know function. I need to know output and input capacitance and delay which you are expecting. Select a circuit family, use static CMOS when in doubt. This has been now proved again in 2008 or 2009 onwards that the best design is easily possible on a static

CMOS at the cost of bit of gate area increasing but the designs are better for performance.

So, if you are specified, you have to work on Domino or you have work on pseudo NMOS or you have to work on zipper that is another issue but if not given always start with static CMOS. Then, sketch the path for the whole network which you are looking for, label each gate with their logical effort, whatever in the series of each path, write G1, G2, G3 and the write equal to their gate paths, each of them has a logical effort.

Label wire lengths and layers, default to minimum pitch. So, if you know the wire lengths, so you will be able to calculate the capacitances associated with wire. Compute wire RC delay because you know the wire dimensions and you also know the wire lengths and which layer, layer means whether the interconnect is on poly or it is interconnected on diffusion or interconnect on metal 1, metal 2, or metal 3.

Whichever layer you are talking you must give the specs for that and based on this R and C can be calculated for wire. So, that is typically RC delay for that blocks, in between blocks you can specify. Why RC delay is reasonable or not, very fine. If they are not reasonable, increase the width and spacing of this or add additional amplifier, buffers there until your delay of wire is attained.

Once you say you are putting a repeater this is essentially a buffer design. Here again you have to use logical effort technique to find how many stages of the repeaters you need in a chain so that the RC delay is of your choice and the delay from that path itself is not larger. Once you know the RC delay of your choice is reasonable, then label notes with lumped wire capacitance, estimate stage effort as I said GHB, compute the gate sizes.

Going back from output towards input, Cin as I kept telling last time, the last stage from the last stage output to input when you go back, the Cin evaluation from your calculation of a bar going from output towards input should actually come as same input which you have started. If there is something wrong, obviously you have to redo the analyses, reduce stage effort again, increase stage effort or add buffers, if anyone of them does not match.

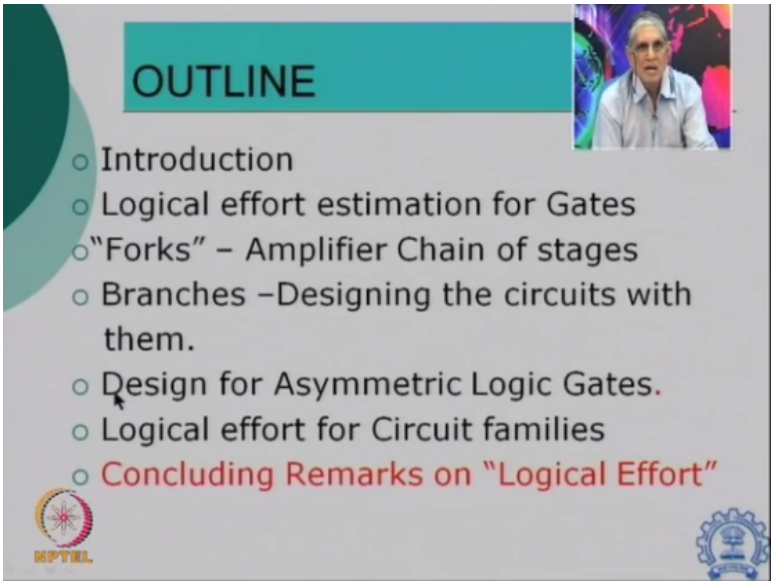
Once the Cin which you are choosing matches of your requirements and specs are given estimate, please remember firstly it should match and secondly it should be sufficient. You already said I cannot have more than this Cin. So, if Cin is less than the specification essentially it means that your stage effort you can reduce because you are unnecessarily putting that additional stage effort.

If you have seen larger, then you must increase the stage effort, so that the current are, as if sizes are higher. So that essentially then it will reduce the value. If your obtained value is much, much larger as I said you like repeaters, put buffers. Till the Cin value is within (()) (01:09:21). Once your Cin value to your specification obtained estimate and simulate delay, if it is again too fast, increase stage effort a bit, reduce Cin.

If it is too slow, can improve the topology and if you improve the topology, go back, scale the new path and do all this analysis once again. If it is not so, then just add what you wanted delay, then you estimate similar delay. Once it is this, you can improve the topology and if you cannot, then you say, here is the catch at the end. If you cannot improve or don't have any other topology, then you say with the kind of topology given the specifications cannot be made.

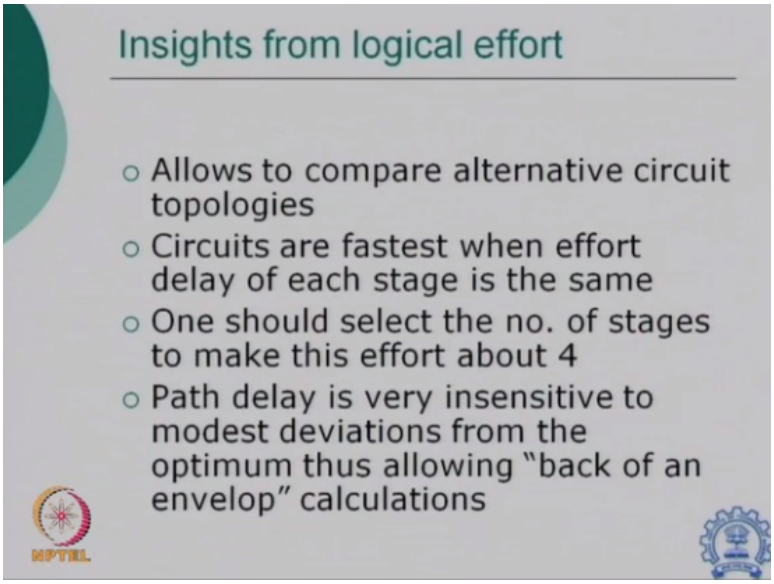
If it is available as I said, you go back and redesign till you are done for the required dealings. This is how high-speed circuits can be designed.

(Refer Slide Time: 01:10:25)



At the end of the day, I may say I may outline now my concluding remarks on the logical effect. Now what is the importance of logical effort I talked so far.

(Refer Slide Time: 01:10:38)



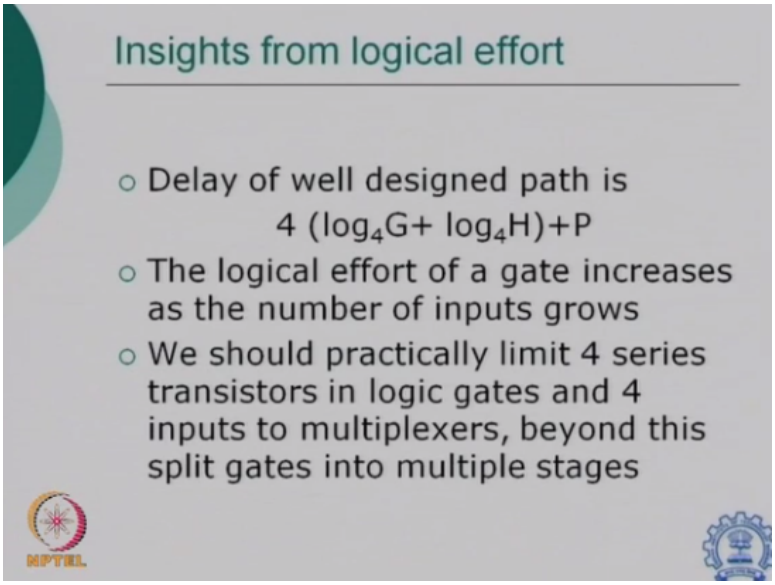
It allows to compare alternatives circuit topology. Circuits are fastest when effort delay of each stage in the same, then one should select the number of stages to make this effort about 4. This is hand bearing calculations, the typical stage effort should be around 4, then you can see that if the stages are too high or low, this effort can be too small or too large and therefore capacitance value will be too higher or too low.

So, for this typical number and not necessary with the exact number. The path delay is very

insensitive to modest deviation from the optimum thus allowing back of an envelope calculation is easy. Please remember path delays are not very strong functions of small deviations and therefore need not worry too much in a normal design and in any case you are going to simulate at the end of this using all other parasitics available.

So, these values are good enough to start with and they may give you a runtime so small that your design can become much faster.

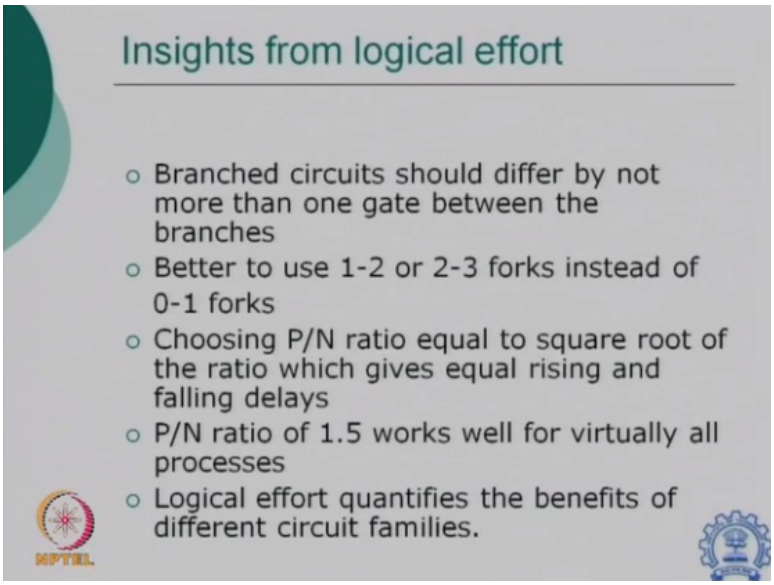
(Refer Slide Time: 01:11:40)



For a given design path, this is an interesting formula which we want to suggest. F is the stage effort which is logically I said is 4 is the best number of stage you should use. Log G to the base 4 and log H to the base 4 x 4 + P is the very design delay path. The logical effort of a gate increases as the number of input grows. This is obvious because the input capacitance is larger, branching is larger, current therefore required is larger.

We should practically limit 4 series transistor in logical gates which is true in any case. If you have too many series transistor, the size of these transistor has to be increased enormously and the power supply may not be able to provide so much current every now and then or split the logics itself into number of blocks.

(Refer Slide Time: 01:12:36)

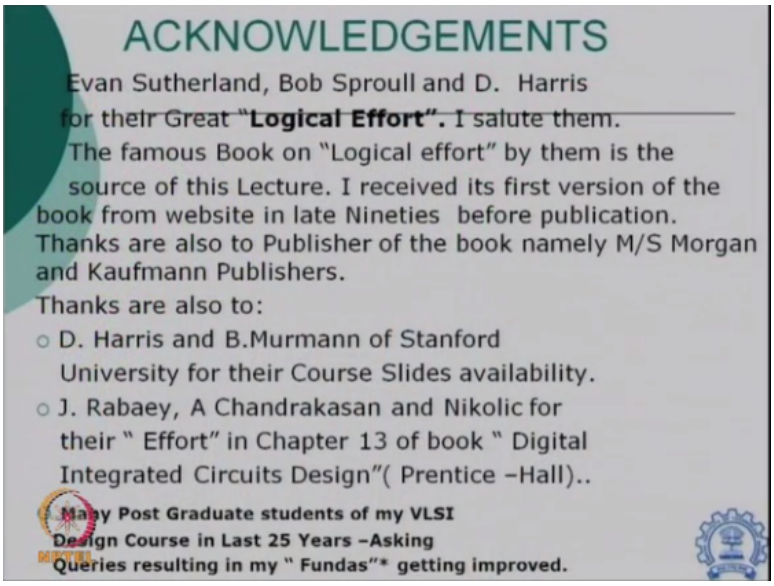


The branch circuit should defer by not more than one gate between the branches. Better to use 1-2 or 2-3 fork instead of 0-1 fork. Please remember, 0-1 fork is always wrong because there is no design on the 0 side and one while never be able to match the 0 delay. So, there is no question of actually saying that I can use one inverter to make clock bar and one direct input as a clock or any signal.

Therefore 0-1 fork can never be used. Only possible to have equal delays are equal efforts is by using 1-2 or 2-3 forks. Choosing P to N ratio equal to square root of the ratio which gives equal rise and fall times, in a normal static CMOS, this is possible. P to N ratio of 1.5 works in many cases, for virtually all processes even up to 28 nanometers. The logical effort quantifies the benefits of different circuit families.

One can actually compare as my flowchart shown you. One can have number of topologies and number of families, one can do logical effort calculations and then verify which of the kind of circuit or families or group of families one can work to get the best delay circuit design. Thank you for all that you heard so far.

(Refer Slide Time: 01:14:05)

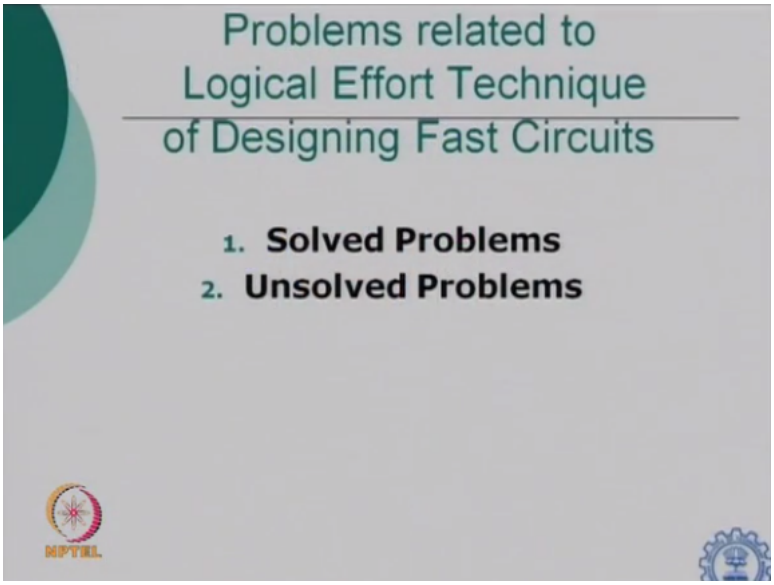


Once again I repeat this acknowledgment for the people who helped us to get to this, even Sutherland, Bob Sproull and D. Harris for their great logical effort. I always say I salute them. The famous book on logical effort by them is the source of this lecture and I said earlier also I received its first version of book from the website in late 1990s before publication. Thanks also to publisher of the book namely Mrs. Morgan and Kaufmann Publishers for published the book.

Thanks are also to D. Harris and B. Murmann of Stanford University for first time giving such quotes and then making available their slides on the website free of charge. Also thanks to Rabaey, A. Chandrakasan, and Nikolic whose book on Digital Integrated Circuits Design is very often used by us as first course in VLSI design. They have a Chapter 13 where they discussed this logical effort relatively briefly if I say in the chapter as section called Effort.

You can also go through it as a first rundown to know what exactly logical effort is and of course I am always thankful to my postgraduate students, at times even undergraduate students of my VLSI design course in last 25 years as they keep asking me very pertinent and interesting queries which actually improve my understanding of many of the areas, particularly the logical effort.

(Refer Slide Time: 01:15:39)



I will give at the end afterwards some solved problems on logical effort and then also give you a list of unsolved problems. Thanks for the day.