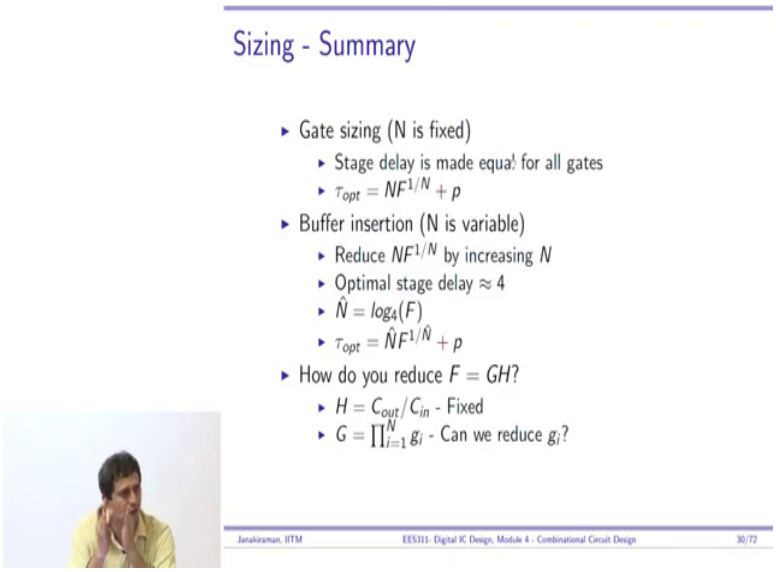
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 44
Input Ordering and Asymmetric Gates

(Refer Slide Time: 00:24)



So, in last class we finished the critical part of gate sizing right. And the summary of gate sizing is as shown here; basically the number of gates in the path is fixed that is N is fixed then the stage delay should be made or the stage effort should be made equal for all stages; you should not have one gate doing more work than the other gate ok.

And therefore, it has to be the net minimum delay that is achievable in that way is N F power 1 by N plus p right, where p is the parasitic effort right. And F is the stage effort where F is

equal to capital G into H path logical effort into path electrical effort which is output capacitance by input capacitance right.

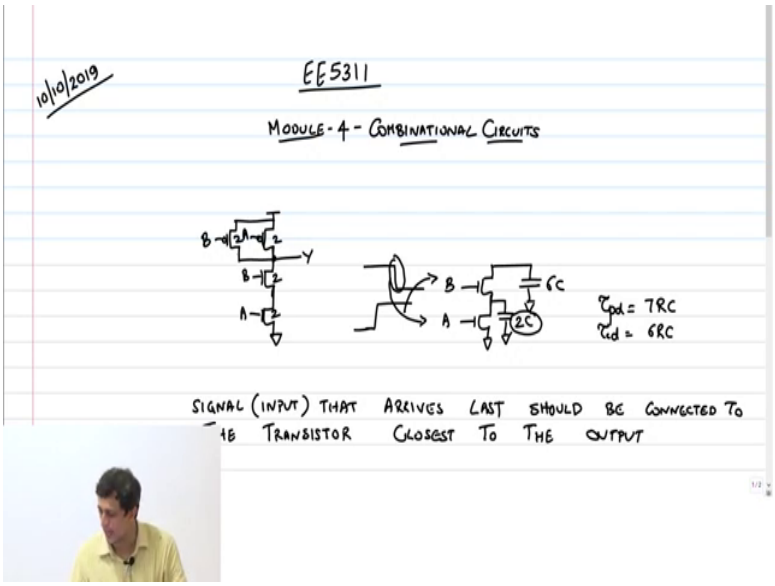
Now, on the other hand it will have very few gates in the path which are driving which is driving a very large load; then this minimum delay is not sufficient you are still going to have lot of effort being put in by each gate right. As you saw in the examples that we did we had an effort of stage effort of 10; now that was too much and we got a delay of 46. Then we said that the reason for that was F power 1 by N is not being made small enough; ultimately that is why your are gaining. So, if you look at this equation N F power 1 by N plus p you have the delay scaling as N right, if N increases my delay goes up linearly with N; but the delay solves that F power 1 by N.

So, these 2 are fighting with each other right. So, you can you must try to reduce F power 1 by N as much as possible without increasing you know until the N starts dominating in the sequence right. And therefore, we said that you can actually insert buffers in the path and still reduce the delay of the path ok. This buffer inversion algorithm was discussed and we said that the optimal stage effort is approximately 4; 3.59 we said making it 4 is not a bad idea because ultimately in your standards say library you will have gates which are of 2 x, 4 x, 8 x, 16 x and so on.

So, that is why even this span out of 4 delay is a very magical number called digital circuits ok. And so, the optimal number of gates after buffering should be log to the base 4 of the path effort F right and this we basically solved and then you can find the optimal delay as N F power 1 by N hat plus p plus N minus N 1 into the delay of inverter that missing here right.

So, further we said that if you want to reduce the delay of your circuit or of your path any further; the only way is to reduce to logical effort because the electrical effort H is simply output capacity by input capacity which it is fixed nothing you can do about it. You are given a load you are given the first gates input capacitance you anyway would not want to upsize that too much you make a unit gate right. So, you take the ratio this is fixed the only way you can reduce the delay any further is to reduce the logical effort of each gate on the path.

(Refer Slide Time: 03:57)



So, in the rest of the module that is what we are going to look at we are going to look at methods in which I can reduce the logical effort ok. So, I before I (Refer Time: 04:05) the logical effort you know let me also be with another important concept, suppose I have a NAND gate here A B and A and B right so, 5 which is 2, 2, 2 and 2 ok. Now, my question is logically these two inputs are identical, I can swap these 2 and still not change the logic of the circuit right.

I cannot do this of course, with A plus B C I cannot just swap any of the input and the logic will change right. Now, when the logic is identical, which input should I feed to which transistor is there a choice are they equivalent or not. And this question also comes in this AB plus B C plus CA circuit that we did right; you have 3 transistors and you have to pick the

transistors and assign the inputs appropriately ok. So, let us look at this and let us look at the pull down path, B and A this is 6 C this is how much of this sorry one second right.

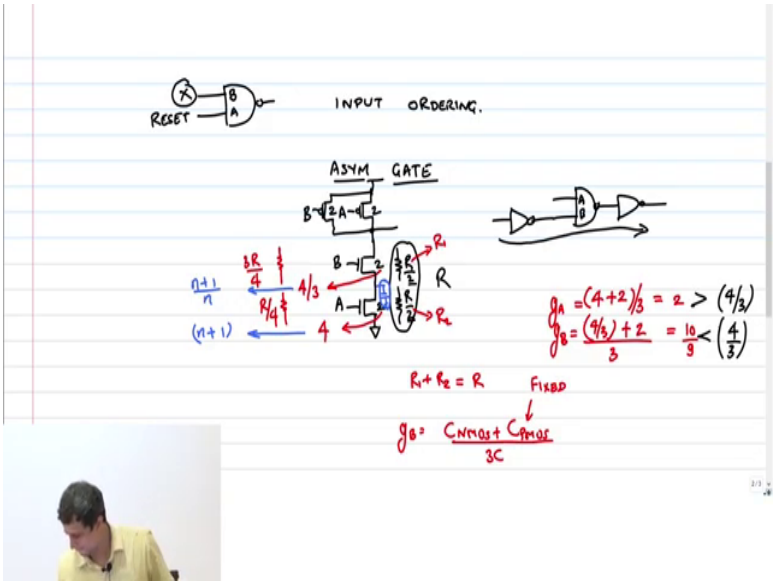
So, we studied this earlier the propagation delay and the contamination delay, we said that the propagation delay p d was 7 times RC; whereas, the contamination delay was 6 RC and the reason for that is this intermediate capacitance here has already being discharged right. Therefore, if I mean now need to only discharged the top capacitor, the 6 C capacitor, that going to be a little faster than discharging both these transistors together.

So, clearly there is and there is a difference in these two inputs right. And therefore, when you have a critical input; means when I say critical input I mean a timing critical input then that input has to be connected to the transistor which is closest to the output ok. So, for example, these two signals A and B right. A might actually switch early but B might switch later ok.

So, when I say A is switching early it means that A has switch to logic high already discharge the capacitor and is now waiting for the input B to switch, when that input comes you want the delay to actually be minimum. So, if I connected the this to B and this to A, then what would happen? The top transistor would have switch to logic 1; it would have sort of charge this intermediately capacitance also to VDD minus VT and it will be waiting.

When the new capacitor I mean when the new input comes and switches, then it still have discharge both the capacities; therefore, the signal which arrives last that is the timing critical signal ok. So, the rule is signal or the input that arrives last what do what do I mean by arriving last? It means that this transition happens later right arrives last should be connected to the transistor closest to the output right. So, this is this statement is true even if you have a NOR gate, if I have a NOR gate to the PMOS stack then the input that arrive last should be connected to the PMOS transistor which is slowest to the output, clear.

(Refer Slide Time: 09:40)



So, what is an example here? Suppose I have a NAND gate, let us say with a signal like this reset and an input A; now this reset is you know very not timing critical right. So, therefore, you connect that guy to the bottom transistor. The signal a is basically the critical signal which is going to control the stuff inside the your circuit right.

So, that has to switch as fast as possible right it will this signal that is going to make many many transitions during the function functioning of the circuit. Reset is occasionally we go low or something like that where you shut off everything and bring the whole circuit to the to a reset condition. So, that is an (Refer Time: 10:29) less critical signal connect that to that transistor which in this case should be A right. So, I want to connect this to A and this is to B or may be I will just call it as (Refer Time: 10:47) avoid confusion I will call it signal X ok.

So, this is input ordering ok. Now, I can take this one step further, input A is very critical; reset is not a critical signal it is going to happen very very very few times ok. So, now, let me see how I can improve the logical effort of this input X right. The question I am going to ask is should this NAND gate have a sizing of 2, 2, 2, 2 or can it have something else where I can help the signal X to switch faster ok.

So, let us look at that careful. So, we said this was B A, A B of course, this it does not matter the would will size is that be have are 2, 2, 2. Now in order for let us assume that this NAND gate is sitting in a path somewhere ok; I have this A and B signal here driving something else I have another inverter here let us. Now, if I want the path delay on this path to be minimum what should I do to the logical effort on this input B? Should it reduce or should it increase? It should decrease, right.

Now, in order for logical effort to decrease should be capacitance on that transistor decrease or increase? It should decrease, right. So, think about it what is logical effort? Logical effort is an is a measure of the capacitance offered for a given pull down resistance or a pull up resistance on the stack. So, the resistance is actually coming from the stack, its not just one from that one transistor.

Because there is one resistance from here and another resistance from here, these two together give me a resistance R ok. And we made this equal to R by 2 and R by 2 in our original sizing where I could not choose between the two inputs; the input were equivalent and in most cases that is how it is A and B are equivalent. Of course, even there if the input ordering I could slightly be preference to one input over the other, but the logical effort of both inputs is still 4 by 3, does not change right.

Now, what I am saying is, I want to reduce the logical effort of this input B further, but I need to ensure where the resistance of the stack does not change. So, what I am saying this R by 2 and R by 2 I am going to make it some R 1 and R 2; such that R 1 plus R 2 is R that cannot change; however, I want to reduce the logical effort of the input B ok.

So, logical effort of the input B is the capacitor of the NMOS plus capacitor of PMOS by 3 C, right this is what it is. If you put it will take the sizing of 2, 2 we will get 2 C plus 2 C by 3 p 4 by 3 ok. Now, the PMOS there is only one transistor I cannot do anything there when the input b switches I need the PMOS to pull up with the resistance R therefore, that has to be 2 I cannot alter that this is fixed. Only thing I can do is now drop the resistance of the NMOS transistors or drop the capacitor I am sorry drop the capacitance of the NMOS transistor ok.

So, instead of 2 I am going to make this sizing as let us say 4 by 3; just some number you can choose the other number also 4 by 3. So, what is the resistance that will be offered by this transistor now? 3; 3 R by 4 therefore, what should the resistance of this bottom transistor be? R by; R by which means this 2 now should become 4.

So, what have we done? We have kept the net pull down resistance be same, but drop the capacitance that this particular transistor is offering to the previous guy and thereby reduce the logical (Refer Time: 16:31) clear. So, what is the logical effort of this of the two inputs now? g A and g B sorry g A and g B. What is g B first? What see NMOS capacitance? 4 by 3; 4 by 3 plus 2 divided by 3 correct. So, what you get here?

10 by 9, what about g A? It is 4 plus 2 divided by 3 how much is that? 2, ok. What all the original logical effort that we had for these two I will put in bracket 4 by 3 and 4 by 3. So, what has happened 2 is greater than this right; however, this is less than this. So, this is called an asymmetric gate because the two inputs are not.

Now, are not symmetric any longer they have different sizes and therefore, different logical efforts right. So, the same example that you take where you have a reset connecting to this; I will not only ensure the reset is connected the bottom transistor and the signal is connected to top transistor I will skew the I will make the gate asymmetric no skew; I will make the gate asymmetric by sizing with like this. Of course, you can now do various kinds of size with this 4 by 3 and 4 is just one example; you can do any N plus 1 by m and N or N plus 1 and get the same result right.

And in the limit; in the limit, the logical effort of this critical input will approach 1; 1 which is as close as you can get to an inverter you cannot get a better than N inverter in any place just statics with of inverter actually you can get better than that. So, I take the statement back we will come to it later, clear any questions here? Yeah.

So, that is the key point that I am trying to make clear see when I say logical effort; logical effort is an is a measure of the capacitance that that transistor offers or that that is that that input will see for a fixed resistance of the stack. The resistance it not coming only from its transistors or whatever transistors that input is connected to resistance is coming from the entire stack right.

Therefore you have these two resistances here, which are contributing to the total resistance. So, what I am doing is, I am ensuring the total pull down resistance will be the same you are right that because I made it 4 by 3 right and not 2; the resistance of that transistor is gone up, but the resistance of the bottom transistor come down. So, what is slightly counterintuitive is that the resistance of the guy you are trying to reduce the logical effort for is actually going up. The resistance of that guy is not coming down it actually going up because I am reducing the capacitance right. So, it does not matter if the resistance of that guy goes up as long as the stack another transistor from the stack is able to compensate that, yeah.

Well, that is not possible right; that is not possible therefore, that is why I said you can make any N plus 1 by N. So, if I make this just think about this theoretically right I can make this as n plus 1 by n and this as n plus 1 right; net resistance if you take you will get R by N plus 1 into N plus R by N plus 1 which is just R right. So, therefore, this resistance will approach this logical effort; if I take N tends to infinitive will start approaching 1, but you cannot do that.

Now of course, practically when I try to implement a layout of an arbitrary n plus 1 by n when n is very large, then the layout itself becomes so asymmetry, that all the assumptions we made would brake, right. This whole thing of ok; there is only one, you know capacitance here there is no capacitance and we know double count that capacitance all that will brake; because the

layout is become more so asymmetric that is why the suggested number is 4 by 3 and 4, that is a practical thing which is going to work even at a layout now. Theoretically you can do any n plus 1 by n ok.

So, what happens to the parasitic delays? So, again the only thing yes so, that will change you got to you know recalculate all that a very little bit and in fact, you will hear you are right that you know the you have to look at the layout; you have to look at the layout now. However, it is still true that parasitic delay does not change, if I upsize will gain that is still remains true; therefore, my gate sizing algorithm still does not care about it.

It is a constant number you have to calculate the new number with the constant number forget about it that is all you have to do one more delay again to get this number right. And in fact, that in between capacitance here you have to recalculate a little bit if may not be the true that with the stack un contacted with each other stuff like that. So, you have to you have to I would just say you have to relay out on commutation skills on the RC extracted net list ok. So, now, so just by the way to this is basically known as an asymmetric gate, where the inputs are not symmetric any longer ok.