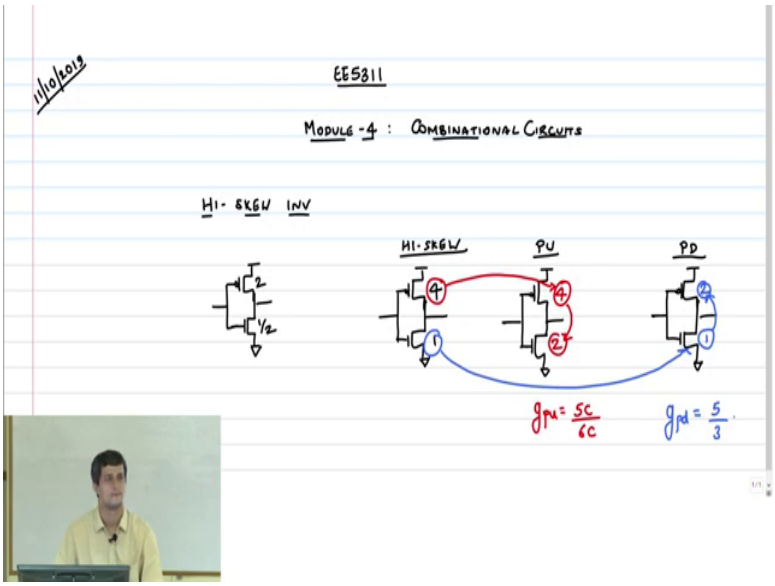
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 46
Special Functions

(Refer Slide Time: 00:14)



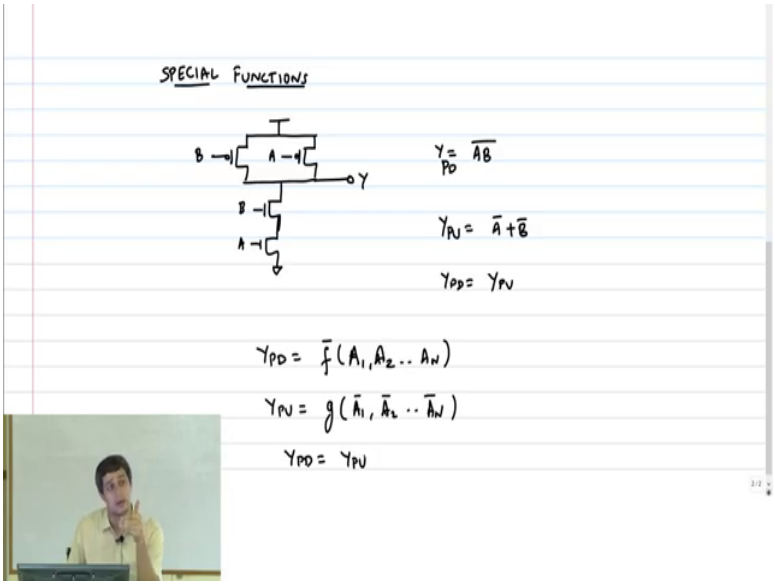
Today, if I take a high skew inverter. We said the size is going to be two and half right. Now some of you had a question what if half is not allowed right, what if half is not allowed in the technology. The point is logical effort is independent of that gate size scaling. So, what I will do is, now I will show you that if whether I take two and half right or whether I take 1 and 4 you will get the same answer ok. So, now, this is my high skew inverter.

Let us say I want to now calculate the pull up logical effort, and this is my pull down logical effort. So, I am going to construct my reference inverters. So, what do I do? If I am looking at

pull up then I ensure that PMOS transistor size remains the same, right. So, this 4 will just appear here. And if that is 4 and I want the inverter to be a symmetric static CMOS inverter the NMOS should be 2. So, this will give rise to the 2 here right. And therefore, what is the logical effort now? g pull up, it is 5 C divided by what? 6 C right. So, you get 5 by 6. Now what about pull down?

This is my reference inverter, what I am going to do this 1 will appear here. Now if this is 1 what should be the PMOS 2? And therefore, I get logical effort pull down equals how much? 5 by 3. This is what we got yesterday right 5 by 6 and 5 by 3. Clearly the pull down is greater than 1; pull up has been made less than 1. So, it does not matter whether I take two half, 1 4, 2 8 all of that is the same ok, clear. Any questions on the high skew inverters or high skew low skew inverters? Ok.

(Refer Slide Time: 03:40)

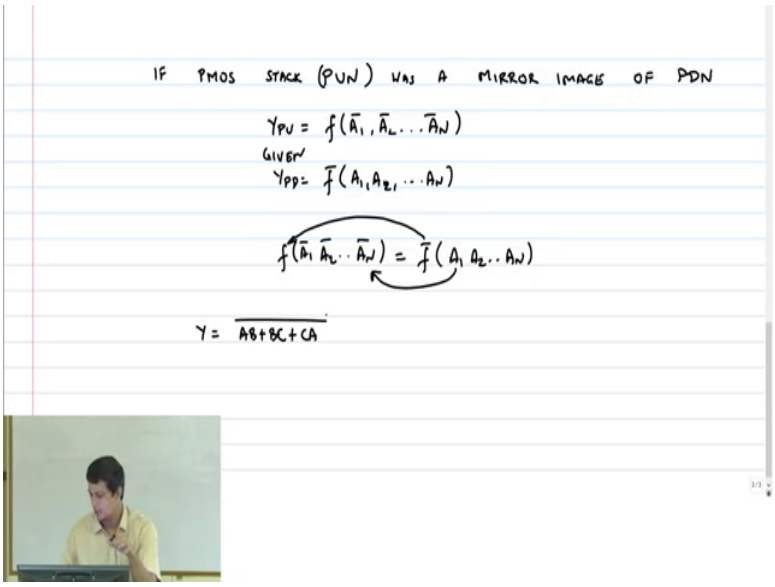


So, today we will look at some special functions ok. What I am going to do is; first let me just quickly review the static CMOS implementation once ok. Let me take LAN 2 again: A B, A and B Y. What is the function that is being implemented in the pull down network? Y equals A B bar: A B bar because its being pulled down to the 0, therefore I have to invert it right; this is pull down. What is the pull up? A bar plus B bar. Clearly Y PD equals Y pull up right. This has to be consistent, ok

In general, if I have a pull down network and a pull up network then I can say that my Y pull down is sum f bar of A B you know or let me take A 1 A 2 A N; f bar because it is going to ground I have to invert that function. And, I am saying A 1 A 2 A N because these are active high switches, the switch is turned on when the input goes high, the general notation is this.

Similarly, if I want to write the pull up network it is some other function g of what: A 1 bar A 2 bar and A N bar. What is the condition? These two have to be the same Y PD is equal to Y PU, both stacks have to implement the same function that is quite obvious right. Now the question is under what circumstances will both these stacks be a mirror image of each other. I want my PMOS stack to be a mirror image of the NMOS stack. Which means that whatever input A 1 is you know connected to an NMOS transistor it is connected to a similar PMOS transistor on top right. So, if the PMOS was PMOS stack was a mirror image of my NMOS stack, ok.

(Refer Slide Time: 07:16)



If PMOS stack right or the pull up network that we say in general was a mirror image. And mirror image because I am taking the mirror image about that x axis going to ground I am make it go to VDD, that is all that is why I am doing mirror image otherwise it is effectively the same stack right; mirror image of the pull down network.

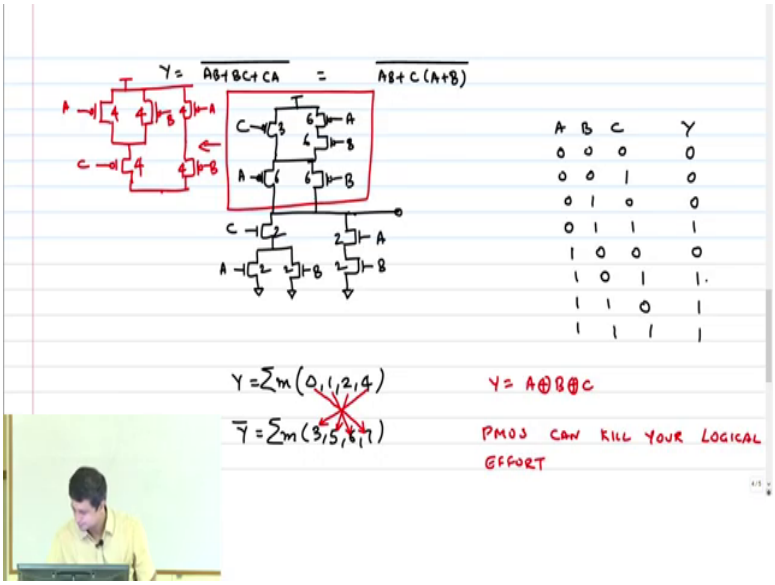
Then, what is a pull up function that is being implemented? Y PU is if the pull down network given Y PD right given is f bar of A 1 A 2 A N. And do all of it, but in terms of f and the A 1 A 1 bars what is it: f of? Exactly this is nothing but f of A 1 bar A 2 bar A N bar. Now we know our other constraint, that the pull down function and pull up function have to be the same.

Therefore, f of A 1 bar A 2 bar A N bar so be equal to f bar of A 1 A 2 A N. What does this mean? It means that, if I complement by inputs A 1 becomes A 1 bar A 2 becomes A 2 bar

and so on then the output also becomes the complement. What I am doing here, these inputs are getting complemented. What this is doing, is output is getting complementing.

So, if by complementing the inputs the output also gets complemented then, I can make my PMOS stack a mirror image of my NMOS stack. And that is what was given to you in the assignment where we said implement Y equal to A B plus B C plus C A whole bar, right. So, what are the advantage of this?

(Refer Slide Time: 10:11)



So, you take Y equal to A B plus B C plus C A whole bar and we do, and let us first do as vanilla static CMOS implementation of this right. Then yeah I will simplify this further, I will just say A B plus C into A plus B. So, this will be C, A B and I have another A and B here, right.

Now what is the PMOS stack? If I, were to do it in the original way where the PMOS stack is just a dual of the NMOS stack. So, A and B have to be in parallel there is a series there. So, I am going to put A B. Then I have C in series with A and B; which means I am going to put C in parallel now with A and B in series, right. So, C, A, B ok. So, can you size this transistor now? So, what should this PMOS sizing be? What is the worse stack size? How many transistors on the stack? 3.

So therefore, I need 6 6 6 right. So therefore, this has to be 6 6 6 and 6 right, and therefore this has to be 3, clear. Now let us check if this condition is satisfied, where I can use the mirroring property ok. So, what is the mirroring property? It simply says that if I complement the inputs output should also become a compliment, ok. So, you can work out this Boolean logic yourself by you are putting A bar, B bar, C bar you worked it out you will get the same function, ok. But I will show you the simpler way which is the truth table way: A, B, C output Y: 0 1 0, 0 1 1. So, what is this? A B plus B C plus C A right. If any two are 1 then we get output as 1: 0 0 1 0 1 1 1, clear.

So, this is what my output Y is mean term of this is A B plus B C plus C A, of course if you want the compliment then its is just a reverse right. So, this is mean terms of you take zeros; 1 0 1 2 4, 0 1 2 4. What is Y bar? Mean term of summation; summation of 3 what 4 5 6 7, 7 ok. This is Y and Y bar. Question now is if I invert the inputs will the output get inverted. So, you look at 0 0 0; if you invert 0 0 0 what do you get? 7. So, you will see that this 0 becomes 7. What about 1? 0 0 1. What do you get if you invert the inputs? 0 0 1 becomes that is 6 exactly. So, this 2; 0 1 0 becomes 1 0 1 which is 5.

And of course, 0 1 1 if you invert you will get 1 0 0 which is 4. So, clearly by inverting the inputs the output is getting inverted. And therefore, I can apply the mirroring property here, clear. So, if I apply the mirroring property the advantage is I can just replicate the NMOS stack. Now the advantage is the NMOS stack has only two transistors on the stack, correct. So, with mirror implementation it would simply this whole thing would become what; just a mirror image of this C A B and this is A B again.

So, what are the, what is the gate size law for this? 4. Is this is 4? 4 and 4. Clearly, the logical effort on each input has come down now, right. This is the advantage. Now, it is not necessary that every time you replicate the NMOS stack is going to be better, in this one example it happened to be like that. First of all there are very few functions where this will hold ok. Another function where it where this holds is the 3 input XOR, if I take Y equals A XOR B XOR C. Then on this again the mirroring property will hold, you can go verify for yourself.

Again, write down the mean terms, see what happens when you inverted: B inputs and C will be output gets inverted right. There are very few functions where this can be done and it can be exploited very well. If that function is important, it turns out this function is very important because this is nothing but that carry out of your full adder. And the three input XOR is the sum of your pull out. And therefore, these two are very important functions in the a l u. And, we are able to exploit this mirroring property beautiful in order to reduce the logical efforts on these three books. Any questions here?

So, the key point; yeah I will come to that later yeah ok. So,. So, the key thing is if your PMOS stack is large and your PMOS size therefore is going to be very large, your logical effort is going to be very high you have no choice right. Basically, the PMOS stack: summary is well I will write it here. Summary is PMOS can kill your logical effort. And lot of effort will now be put in in order to optimize this PMOS stack. One such technique is this mirroring property, ok.