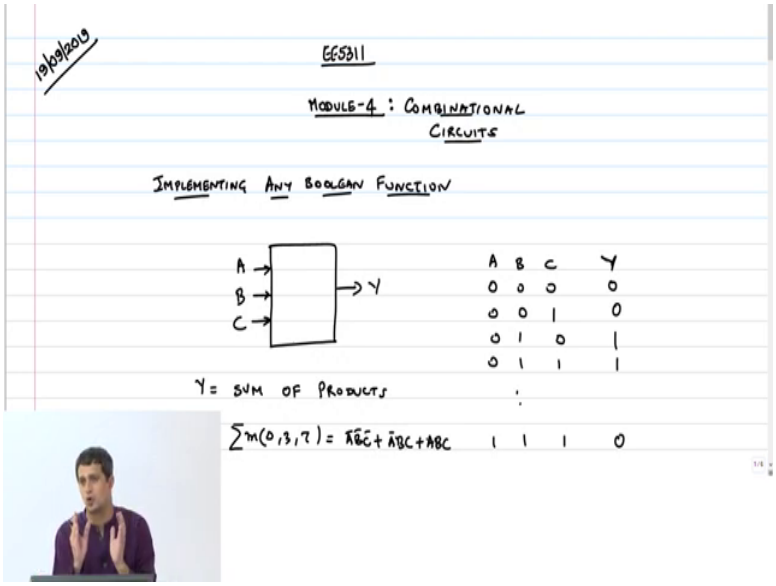
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 33
Implementing Any Boolean Logic Function

So, let us move on our discussion on combinational circuits right. So, last class we started discussing very briefly the idea of implementing any arbitrary Boolean function right.

(Refer Slide Time: 00:29)

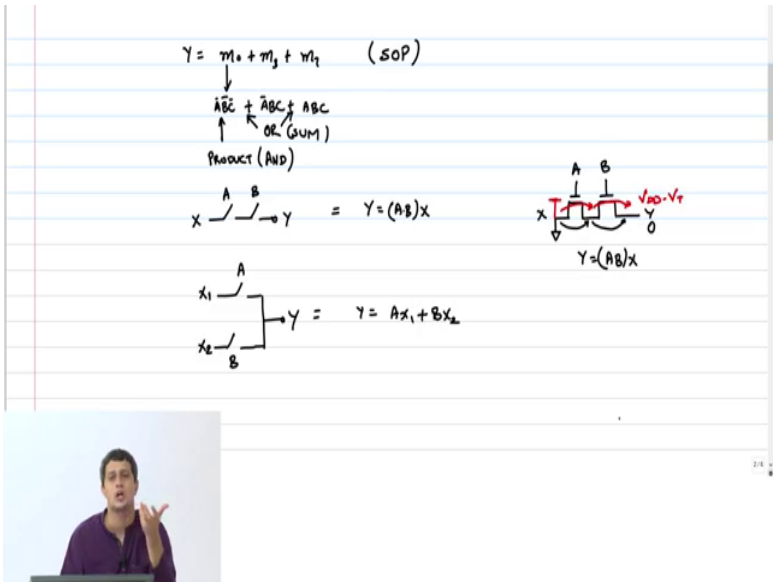


Implementing any Boolean function; so, the idea here was I have a black box with inputs let us say A, B, C, A, B and C and some output Y. Let us just take one output. So, therefore, you write down the truth table for this for 0 0 0 0 0 1 0 1 0 1 1 all the way to 1 1 1. And, then you

put down the locations where the output will go to 0s and 1s right. This is how you solve the K-map right.

So, in summary you can represent the output as a sum of products. As a sum of products and what is that, that is basically also known as minterms summation of minterms or let us say 0 3 7 for example right. So, what does this mean? 0 is A bar B bar C bar. 3 is A bar BC and 7 is ABC ok. Of course, you can simplify this by you know combining the A plus one bar and make it one and all that, but let us not get into that let us assume that you simplified this whole thing and then you have some expression that you want to implement.

(Refer Slide Time: 02:31)



So, what does this sum of products mean right. I have a minterm is 0 plus minterm is may be 3 plus minterm is 7. Each minterm is a product of the inputs, some product of the inputs. Like

here it is A bar B bar C bar right then I have an or of the other term A no A bar BC plus ABC right.

So, there is a product right and that is nothing, but an AND condition and there is a sum operation that is being performed here which is an OR condition this is the SUM, that is why it is called sum of product SOP right. The idea is now to take this and implement it in static CMOS logic ok. So, the point is if I have any arbitrary switch right and I put two of these in series controlled by signals A and B then the signal X rub here will go to the output Y when both A and B are closed right. So, this can be written as Y equals A and B into X.

So, there is a Boolean operation on A and B and then it is passing the logic value X to the output Y right. So, similarly if I have 2 arbitrary switches which are in parallel right and I have input X1 and X2 right then my output Y would now be a times X1 plus B times X 2 right. Now we have 2 kinds of switches in static CMOS logic that we can use. One is an NMOS transistor and other is the PMOS transistor right.

So, what we said last time was if I have an NMOS transistor like this right trying to pass the logic X then Y will be AB into X ok. The only catch here is that we already know that if X were grounded then there is absolutely no problem and if the signals A and B go high which means they go to V DD then both NMOS transistors will turn on. The 0 from here will appear here and that 0 will also percolate to the output Y. No problem. On the other hand if I connect this to V DD then what will go through from here to here is what.

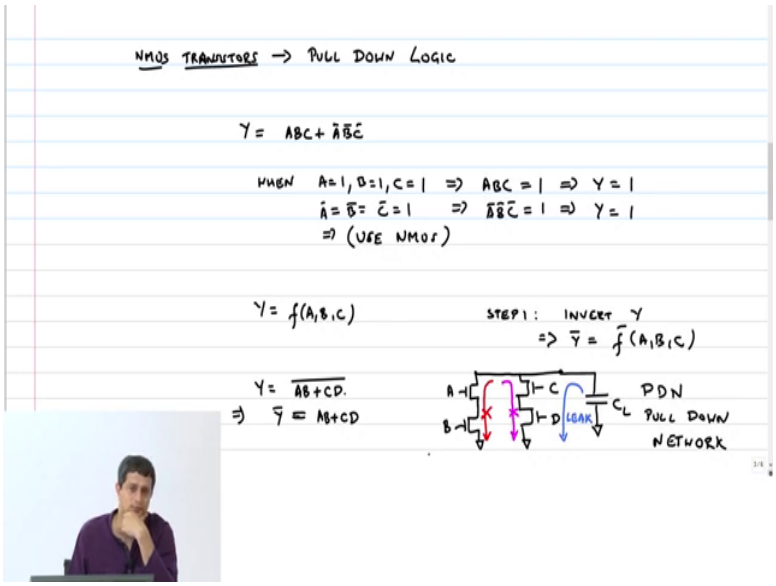
Student: V DD minus V T.

V DD minus V T and further from here to here is also.

Student: V DD minus.

V DD minus V T right where as for 0 it goes to ground right. And therefore, the problem with using this NMOS switch is it is not equivalent for both logic levels right.

(Refer Slide Time: 06:16)



So, the conclusion is that the NMOS, NMOS gate transistors actually. NMOS transistors can be used only for pull down logic. It can be used to pull the output loop in some cases ok. In other cases the NMOS transistor will not help because they cannot pass a logic 1 fully. You cannot get the rail to rail output with logic 1.

Clear. So, let us go back now and look at this sum of product term that we had a let us say ABC plus A bar B bar C bar. So, what does this thing mean. It means that when any of the minterms become 1 right then the output is going to go high and all the switches in your in each of the minterms is an active high switch right. So, when A equal to 1, B equal to 1 and C equal to 1 implies ABC equals 1.

This implies Y equal to 1 right or if A bar equal to B bar equal to C bar equals 1. This implies A bar B bar C bar equals 1 implies Y equal to 1. So, the inputs that are going to control the

switches are active high inputs and when any of these minterms get activated the output has to go high right. Now, this is so, this active high inputs is not a problem because naturally the NMOS transistor has an active high input.

When that input goes high a goes high the NMOS transistor will turn on and therefore, the output can the switch will turn on basically right. The only problem is passing this logic 1 to the output when the switch is turned on. So, the SOP assumes that when any of these minterms go high the output has to go high right. Unfortunately, the NMOS transistor cannot pass a logic high right. Why are we using NMOS transistor?

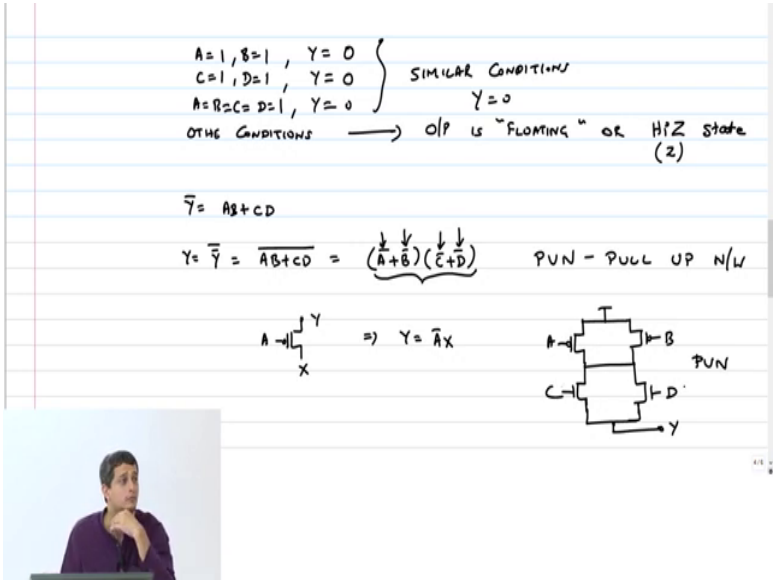
Because the switches are active high switches here. In this SOP thing switches are active high switches and therefore, we are going to say use NMOS. Unfortunately, you cannot use it to pass a logic high as the SOP normally demands. Therefore, what you have to do is you have to first invert the function if you want to implement any arbitrary Boolean function in static CMOS logic.

So, let us say I want to implement f of A, B, C. The first step 1 is invert Y right. You write Y bar equals f bar of A, B, C and go ahead and implement this f bar logic in and for the NMOS gate first. We look at what we should do for the PMOS in some time right because I want to get the output to go to logic 0 with the NMOS, but only if I invert will actually logic will be correct and that is why static CMOS logic is inverting by nature right. So, you can go head and use this idea that I told you here where if you put an 2 NMOS transistors in series or n NMOS transistors in series you get a and condition.

If you have an OR condition then you have to just put the n in parallel ok. So, let us look at an example. I will look at Y equals AB plus CD right. So, the first thing I have to do is I want to implement Y bar first. Y bar is AB plus CD. No I will actually consider the compliment. So, that is easy right then Y bar will become AB plus CD now I have A into B that has to be implemented using NMOS transistors.

So, what is the; I put 2 NMOS transistors in series; so, AB. Now I have the C into D now which has to be implemented C and D and these 2 now should be in parallel. So, therefore, you stop this output and you get output Y right.

(Refer Slide Time: 12:36)



So, when A equal to 1, B equal to 1. The output Y is 0 when C equal to 1 and D equal to 1 output Y is 0 right. What happens in the other cases? In the other 14 combinations, what will the output Y be?

Student: Time B.

Yeah.

Student: Time B

Yes essentially what is happening is when any of this pull down paths turn then output is pulled low and it is held low right. So, if you look at this. If A and B turn on this pulled down path is activated the output Y is simply held low as long as A and B are high. If C and D turn on then this pull down path is activated right and the output again is held low as long as C and D are on. Of course, if A, B, C and D are on then you have 2 paths both of them pull down to 0 right.

So, I was wrong in saying other 14 cases because it must be other 13 cases right because A, B, C, D equal to 1 also is 0. What happens in the other case when these conditions are not satisfied is you have capacitor here right. Some parasitic capacitance load capacitance all of that right. You have capacitance here which is storing some value initially based on the previous state. Now suppose the inputs go into one of these other 13 conditions; that means, there is no pulled down path through any of these. The input is in that kind of condition then the output on the capacitor is just held as it is.

And this is known as a high impedance state. Why? Because, there is a high impedance from the capacitor output to any of the rails and therefore, the capacitor voltage is suppose holds it is value there for actual till infinity if there is no leakage current right. Unfortunately, if A and B for example, they are connected to the real transistors then there is going to be some threshold leakage.

So, then this capacitor will leak out slowly like this, but that is very slow. So, for all practical purposes you can assume that the capacitor will float at that particular value for long enough right. So, A equal to B equal to C equal to D equal to 1 also Y equal to 0 other conditions yeah.

Student: What about the condition 1 1 0 0?

Yeah 1 1 0 0; so, no sorry can you tell me again.

Student: AB (Refer time: 09:29) can be 1 1 0 0 is of other thing.

AB can be 1 1 and 0 0 of the other things yeah.

Student: (Refer time: 16:11).

Ok. So, that is right.

Student: (Refer time: 16:14).

So, right ok. I will basically let us not get into that. Similar conditions right Y equal to 0. Other conditions output is floating ok. You will hear this term or high impedance in high impedance state right. And we basically represent this by Z. So, typically high impedance means the output is going to float on a capacitor somewhere ok. So, we will we will use this at a later point in our right. So, what are we done now. We had a function f of A, B, C, D inverted it we implemented the pull down stack right.

This basically is the pull down network pull down network pull down network. Now what about the pull up network? In certain other conditions I want the output to go high right. So, that I have not yet done. So, if you take my thing where I inverted I got f bar of A, B, C right. Then I now need to do something in order to implement the pull up pull up stack right.

So, what you do there Y bar equal to AB plus CD right. So, if I now complement this. The output that was supposed to go 0 will go high right. So, now, I will write Y equal to Y bar whole bar which is basically our original function right. I will tell you how to do this in a simpler way in a minute. I am just showing you formally what has to be done.

Now, this can be written like this use De Morgan's laws. Make all the ANDs ORs all the ORs ANDs and pull the compliments inside right. So, it will become A bar plus B bar into C bar

plus D bar right. So, there is no way that any of these combinations can cause the output you know the pull up network also to turn on and the pull down network turn on because we have complemented these 2 right.

So, there is no pull down path or there is no path for the current from V DD to ground in any of the input combinations right by design that is what has happened. So, now, we need to go ahead and implement this function in our pull up network ok. This is for the pull up network. Now remember I told you that a PMOS transistor the switch is an active low switch. So, if I connect an input a like this and this is some X Y it implies Y will be A bar into X because A has to go down to 0 assuming that the V DD on some other node right which will make it the source.

The transistor will turn on and there by conduct right. And therefore, this is an active low switch. So, when you see an A bar you actually just need to connect A to that PMOS transistor. Here look at this. This one, this one and this one right. So, what you do for the pull up network you are now going to. I have 2 terms A bar plus B bar using 2 PMOS transistors and that will be a parallel or series combination.

Student: Parallel combination.

It will be a parallel combination A bar plus B bar A bar plus B bar and this is going to go be V DD. Now that is going to be in series with an other network C bar plus B bar right and this is my output Y. Again here the output Y when the input combinations allow a path to V DD right which could basically go through any of these combinations right. The complimentary combinations of when it would go to 0 the output Y would go to 1 and otherwise it could be in high impedance because I not yet connected the pull down network here.

I have just connected a pull up network to V DD. So, in other conditions the output will float ok. So, the first useful lesson here is when two outputs are floating in a complementary manner right. When the output is 0 in the pull down network it goes to 0 otherwise it floats. In exactly those conditions that guy pull up network is going to pull it up to V DD. So,

therefore, I can tie these two outputs to a weird or connection on these 2 things and get a proper output right; so, to implement this function Y equal to AB plus CD whole bar.

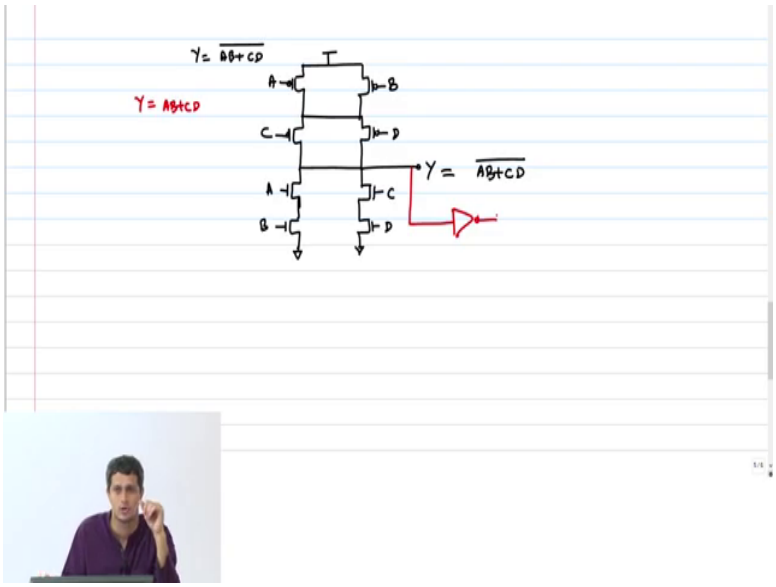
Student: (Refer time: 22:46).

Yeah, that is what sorry, sorry, sorry, sorry mistake, mistake, mistake thank you A, B, C, D you are right.

Student: (Refer time: 23:03) PMOS transistor.

Yeah this must be a PMOS transistor. Yes correct A, B, C and D ok.

(Refer Slide Time: 23:22)



So, now I just put these 2 pull up network and pull down network together I get my Boolean logic function that I wanted ok. So, I have A, A, B, C, D, C and D again. I am shorting these 2 A and B. This function is AB plus CD whole bar ok. Now what if, I asked you to implement Y equal to AB plus CD. Then you have to invert it implement this AB plus CD whole bar and then add and inverter on the path which of course is just one PMOS NMOS transistor and you get it.