

**Microelectronics: Devices to Circuits**  
**Prof. Sudeb Dasgupta**  
**Department of Electronics and Communication Engineering**  
**Indian Institute of Technology Roorkee**  
**Module 11**  
**Lecture 53**  
**Combinational Logic Design -I**

Hello everybody and welcome once again to the NPTEL online certification course on microelectronics: devices to circuits. Till the previous module, we had covered the basic of devices and the usage of devices for analog circuits. So primarily, till the previous module which is about week 6 or week 7, we have actually dealt with the whole of analog design. The most, at least the basics of analog design and so that given a MOSFET, either an N channel or P channel, now you will be in a position to actually design for example basic filters, amplifiers, differential amplifiers, operational amplifiers, you would also be in a position to understand the device physics as we have understood in our previous module.

From this module onwards, we start our journey for digital logic design. This we will keep it short, so about say about 4 hours will be devoted for digital logic design because the same has already been done in your previous modules. So we will be repeating the slightly varied amount, keeping in view your requirements for this course. So we will start with today's module or talk on combinational logic design and this is part 1 module of the combinational logic design.

(Refer Slide Time: 2:08)

**Outline**

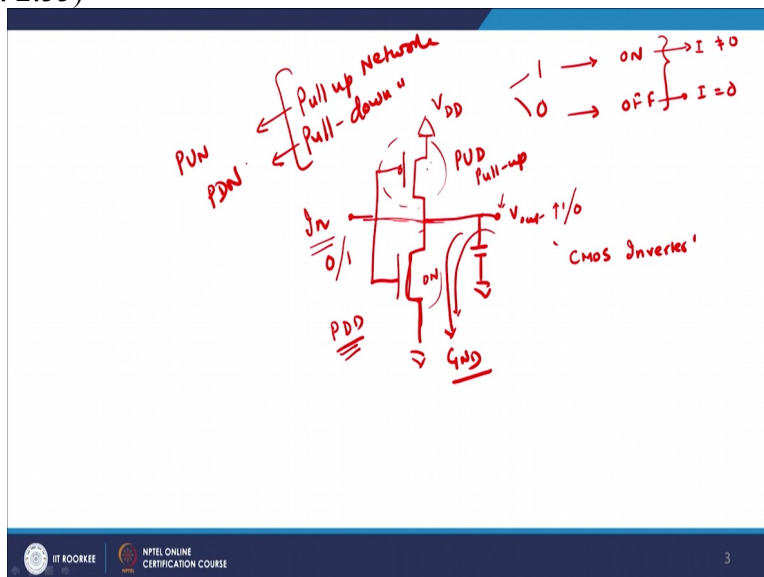
- Introduction
- Static & Dynamic Logic Design ✓
- Choice of Pull-up and Pull-down Network }
- Two Input NAND Gate }
- Static Properties of Complementary CMOS Gates } →
- Propagation Delay of Complementary CMOS Gates }
- Problems of Complementary CMOS Gates }
- Design Techniques of Large Fan-in }
- Recapitulation

IT Roorkee | NPTEL ONLINE CERTIFICATION COURSE | 2

So what we will be doing in the next half an hour or so is, give you an idea about what is combinational logic design, we will have a look into what is known as a static and dynamic logic design. So this will be, we will be looking into the static and dynamic logic design, right, what is the meaning of that and then what makes your device go ahead and choose pull up and pull down network.

So which device will you use for pull up networks and which device should you use for pull down networks? We will explain each one of them. Then we will take an example of a two input NAND gate and then we will go into static properties like for example, complementary gate and then we will look into the propagation delay and then basically designing of large fan-in circuits and then we will recapitulate the whole thing. So this is the basic flow which will happen.

(Refer Slide Time: 2:55)



Now just to give you an insight of what we are trying to do here is, that till now we were assuming that you have an input which is basically a sine wave and you are supposed to know how does the MOSFET or the active device behave at each and every point of the input cycle which means that if the device is triode region, saturation region, active region and so on and so forth. You were also told that the current equations for each region should be known to you right.

In digital logic design whereas, you actually work with only 2 phases, which is switching on and off. You must be knowing that in digital logic, you only work with 1s and 0s. So this corresponds to on state and this corresponds to generally you know off state right? And therefore

there will be only 2 states and therefore in this case there will be some current flowing, in this case the current will be equals to 0 and this case the current will be equals to nonzero, right.

So that makes my life relatively easier when we are dealing with digital logic. We have already seen in our previous turn that whenever we are discussing a CMOS transistor, if you have an NMOS and PMOS then when you give, say this is your input right and this is your output,  $V_{out}$  and you have a load capacitance here. And this is your  $V_{DD}$  then when you given input here which is equals to say 0 then this switches on and  $V_{out}$  goes to 1, right?

When this becomes 1, the NMOS switches on right and this voltage here drops down to 0 and therefore this goes to 0. So and this is known as a basic CMOS inverter, right? This is inverter. Now in this case, as you can see, we can divide the whole thing into 2 parts. The top part and the bottom part. So this is known as pull up device, PUD and this is known as a pull down device, PDD. So this is pull up device. Why pull up? Because this device helps you to pull up the voltage at this particular point to  $V_{DD}$ .

And why pull down device? Because this device tries to pull down the voltage at this point down to ground. This is your ground, right. Now if you have combinations of PMOS in the pull up, then we define that to be as the pull up network. If there are combinations of PMOS<sub>s</sub>. And similarly, if you have combination of NMOS<sub>s</sub> design at the bottom, then we define this to be as a pull down network. So we refer to this as PUN and this as PDN right. This is the basic fundamental principles based on which we will be starting to look into our whole issue of combinational logic.

(Refer Slide Time: 5:43)

Introduction  $o = f(i)$

- In the combinational logic or non-regenerative circuits the output at any instant of time depends only on the signal present at its input.
- In contrast to this if the output depends on the current input data along with the previous state of input then it is called as sequential or regenerative logic.

Inputs → [Combinational Logic Circuit] → Output

Inputs → [Combinational Logic Circuit] → Output

↓

States

↑

[Combinational Logic Circuit]

Inputs → [Combinational Logic Circuit] → Output

IT BOONKEE NPTEL ONLINE CERTIFICATION COURSE 3

Let me come to the combinational logic, first of all give you an introduction of combinational logic. Combinational logics are the most simplest circuits of digital logic design and as I discussed with you or I will be discussing just now, is that in a combinational logical circuit if you look at this particular diagram here, you see I have got a set of inputs here on the left-hand side and I have a set of outputs here on the right-hand side right?

Now if you look at this 1<sup>st</sup> bullet which you see, in the combinational logic right, we will not discuss at this stage non-regenerative but that is in the combinational logic, the output at any instance of time depends only on the signal present at that input which means that depending on the inputs available here at a particular instance of time, the output is determined, right. So if you can write down, output is actually only function of inputs at that particular time. It does not depend upon anything else, right?

Whereas, if the output which is the 2<sup>nd</sup> bullet statement, if the output depends on the current input data which is this one, along with the previous state of the input, we define this to be as a sequential or regenerative logic which means that if the output from the previous sets of input helps you to determine the current state, we define that to be the sequential logic. So what is the primary difference between sequential logic and combinational logic?

Combinational logics are responsible for giving you an output based on current inputs. For example NAND gate, NOR gate, XOR gate, Ex- OR, any of the standard gates available to you

whereas, a sequential logic output not only depends upon the present state of input, it also depends upon the previous states right? And therefore you have a memory sort of memory here because you are storing at least some amount of data for a finite amount of time.

Therefore, these are also known as sequential logic, also known as regenerative because you are regenerating the output based on the previous inputs whereas, the combinational logic is basically non-regenerative. And therefore combinational logics are also referred to as non-regenerative circuits whereas sequential logics are referred to as regenerative logics and circuits, fine. So, combinational logic therefore let us understand once again, it depends only on the inputs in the present states.

Sequential logic depends on the present state inputs as well as remember, previous state inputs right, on the previous outputs. Now, what we will be looking into this about 2 modules or so or 3 modules or so, we will be concentrating on the combinational logic itself right. Within the combinational logic, we have 2 types of logic once again. One is known as static and another is known as dynamic logic right.

(Refer Slide Time: 8:37)

Static & Dynamic logic Design

- Static Design-At every point of time, each gate output is either connected to  $V_{DD}$  or  $V_{SS}$  via a low resistance path.
- Dynamic Design-This relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes.
- Complementary Logic- This is a widely used static logic, consists of pull-up and pull-down network (PDN).

The diagram illustrates a static CMOS gate structure. It consists of two main blocks: a PULL-UP NETWORK (PUN) and a PULL-DOWN NETWORK (PDN). The PUN is connected to the supply voltage  $V_{DD}$ , and the PDN is connected to the ground voltage  $V_{SS}$ . The output of the gate is labeled  $F(I_1, I_2, \dots, I_n)$ . Handwritten green annotations include:  $I_1 \rightarrow I_n = 0$  near the PUN,  $Z_{out} = 0$  near the output node,  $Z_{in} = 0$  near the PDN inputs, and 'gate %' near the PDN. The slide footer includes the IIT ROORKEE logo and 'NPTEL ONLINE CERTIFICATION COURSE'.

Now static logic primarily means that at every point of time, each of the gate output which is this one, this is the gate output, right is either connected to  $V_{DD}$  or  $V_{SS}$  via a low resistance path.

I suppose you can understand. So let us suppose your all your inputs here,  $I_1$  to  $I_n$  are all 0, right and suppose all the PMOS<sub>s</sub> are in series here right. Then all the PMOS<sub>s</sub> will be switched on, so therefore I will have a low resistance path between output and  $V_{DD}$  and this  $V_{DD}$  will appear on the gate side, right. Similarly, if all your inputs are 1 here and all your NMOS devices in the pull down network are in series, then all your devices will be switched on together and you will have a low resistance path from your output to the  $V_{SS}$  right.

So you will have a low resistance path at this particular point right. Then we define such a design to be a static design, right. So what is a static design? Whenever the output is connected to a low impedance load or is connected to a  $V_{DD}$  or  $V_{SS}$ , which is output and it is not floating therefore, please understand, it is not floating. It is either connected to  $V_{DD}$  or  $V_{SS}$ . So this is therefore, the impedance at this particular point,  $Z_{out}$  will be approximately equals to 0, right.

So output impedance is 0 and input impedance is infinite because, why input impedance is infinite in static design or for that matter, any CMOS structure is because you are sending the signal onto the gate side. The gate is always separated from the channel by oxide layer and therefore there will be no current flowing in this direction. So  $V$  by  $I$  will be infinitely large and therefore your input impedance will be infinitely large. So in ideal cases,  $Z_{in}$  equals to infinity and  $Z_{out}$  equals to 0, right. So this is your static design.

What is a dynamic design? We will look into dynamic design later on but just to give you a definition in thoroughness. This relies on the temporary storage of signal values on the capacitance of high impedance nodes, right? Now this is quite interesting that in a dynamic design, you hold the value of the voltage at the output node through  $C_L$ , right till the time when you are not changing, means you need to change it externally, right. So you will be storing some amount of the time the output side, right.

When you are storing the data, it has to be minimum stored till a point till your new sets of inputs do not arrive, right. So typically, it is a floating node sort of node in a dynamic operation. Now the design which you see in front of you in this slide is basically known as a complementary logic, right. This is widely used static; the most widely used static logic design is the complementary design, as you can see here. Complementary means, you will have PMOS<sub>s</sub> in the pull up and NMOS<sub>s</sub> in the pull down and complementary why?



Because if you give input high, output will be low; if you give input low, output will be high. So they are complementary of each other, a basic CMOS static. So what we have learned across this, static where output node is connected to either  $V_{DD}$  or to  $V_{SS}$ , having your output impedance almost equal to 0, dynamic is a floating node and where the charges and the voltage is stored in the output side till a new set of inputs are not given to it. Most of the time, we will be discussing about the static design only, right. Okay.

Let me see why should we therefore choose a pull down or, what is the reason for choosing pull up and pull down network? Now, I have been telling you that all my pull up networks should always be made up of PMOSs and all my pull down networks should be always made of NMOSs and there is a specific reason why we do like that, right. The reason is something like this and I will tell you the reason in front of you. Say for example, your, this is your NMOS right.

(Refer Slide Time: 12:49)

**Choice of Pull Down Network**

- What type of MOS is preferable for PDN?



5

This is your NMOS, let us suppose and you give a high-voltage  $V_{DD}$  from the gate side. So what will happen is, this will switch on and the output voltage here will be dragged to 0 and therefore  $V_{DD}$  goes to 0, right. There is no problem at all.  $V_{DD}$  will go to 0 and what happens to your, finally what happens? Your  $V_{DS}$  becomes equals to 0 because this is grounded. So  $V_{DS}$  is equals to 0. Why? Because source is grounded, drain is also grounded, finally, you get  $V_{DS}$  equals to 0 and  $V_{GS}$  equals to  $V_{DD}$ .

So when  $V_{DS}$  equals to 0, finally I will get no current and I will get output equals to 0 and this is stable condition. Let us put a PMOS here and let us see how it works out. See, what happens in this case is that suppose you, because to switch it on, you have to ground it, right because PMOSs require grounded to be switched on, NMOSs require a positive voltage to be switched on because of threshold voltage.

If you apply a bias here which is grounded, this is switched on, agreed with you and therefore the voltage at this point will start to fall down, right? Who is storing the voltage? This capacitance is storing the voltage in both the cases. So this voltage will start to fall down but here comes the big issue that just as the voltage here, right becomes equals to mod of  $V_{TP}$ ,  $V_{TP}$  is the threshold voltage of the device, this device goes into cut-off. Right?

And that is the basic problem area of a PMOS that, whenever my  $V$  out reaches approximately equals to mod of  $V_{TP}$  right, this ensures that this is switched off. And as it switches off, you will automatically, you can understand the reason why. Because if this is mod of  $V_{TP}$  the difference between these 2 is mod of  $V_{TP}$  but understand, for device to be in the on state,  $V_{GS}$  should be greater than equals to the threshold voltage, right? That was what my basic definition is all about.

So gate to source voltage is 0 but I have a mod  $V_{TP}$  here which is basically a fixed value. It cannot be, never be greater than mod of  $V_{TP}$  and at mod of  $V_{TP}$  this switches off and as it switches off, my output voltage latches to a value equals to, just equals to mod of  $V_{TP}$  which means that my output voltage is not able to go directly to 0 but it only latches to a value equals to mod of  $V_{TP}$  which means that let us suppose the PMOS has got the threshold voltage of 0.5 volts, then rather than the output going to 0, my output will actually fix to 0.5 volts.

So I will not be able to get the whole swing from  $V_{DD}$  to 0. That is the reason we use a pull up, we use a PMOS for all practical purposes. Similarly, let us come to pull down network.



(Refer Slide Time: 15:46)

### Choice of Pull UP Network

- What type of MOS is preferable for PUN?

$V_{GS} = \frac{V_{DD}}{2} - (V_{TN} - V_{TN}) = V_{TN}$   
 $V_{GS} = V_{TN}$

$V_{DD} = 1.8$   
 $V_{TN} = 0.2$   
 $V_{GS} = 1.6V$   
 $V_{GS} = 0.9$   
 $V_{GS} = 0.1V$

HIT BHOORKEE    NPTEL ONLINE CERTIFICATION COURSE    6

Sorry, in a pull up network. We were doing pull down, we will do pull up now. In pull up, let us suppose, I have a PMOS right, I have a PMOS grounded again, this is connected to  $V_{DD}$  right? There it was 0, here it is connected to  $V_{DD}$ . Now my, this is initially 0, this is switched on. This voltage rises to  $V_{DD}$ , right and goes to  $V_{DD}$ . Now as it goes to  $V_{DD}$  right, the idea is that this will actually, can eventually go up to  $V_{DD}$ . Why? Because then the  $V_{DS}$  will be equals to 0.

And that will ensure my device will be switched off and therefore it will be switched on only at  $V_{DD}$ . So which ensures that the PMOS will actually raise my voltage at the output side to  $V_{DD}$ . So there is a full swing available to me. Whereas in this case, if we take NMOS, then when this is switched on and I have 0 here, if it is switched on, again a low resistance path is there and therefore this voltage will start rising. It will rise, agreed but it will rise to a value equals to  $V_{DD}$  minus  $V_{TN}$ . Why?

Because if this is  $V_{DD}$  minus  $V_{TN}$ , right and then  $V_{GS}$  if you find out, this is equals to  $V_{DD}$  minus of  $V_{DD}$  minus  $V_{TN}$  which is nothing but  $V_{TN}$  because this will get cancelled out. So I get  $V_{GS}$  equals to  $V_{TN}$ . Now if the voltage rises above this, then  $V_{GS}$  becomes less than threshold voltage of the device and this device is switched off. Exactly the same as I discussed in the previous slide. Are you able to get the picture?

See, so the idea therefore is the output has to go from 0 to  $V_{DD}$  but as it goes to  $V_{DD}$  minus  $V_{TN}$ , this device which is NMOS here, will be cut off, will go to cut off because your  $V_{GS}$  becomes

less than  $V_{TH}$  as the output voltage goes above this and therefore the output voltage, this voltage latches to  $V_{DD}$  minus  $V_{TN}$  and which means that if I use a NMOS in the pull up network, my output will be only latched to  $V_{DD}$  minus  $V_{TN}$ .

So if you say,  $V_{DD}$  is 1.8 volts and  $V_{TN}$  is 0.2 volts, then I will be only latched to 1.6 volts and I will not be able to go to the whole swing attached to me, right? We will see later on that if you are not using a whole swing, there is some problem, right and you are very close to  $V_{DD}$  by 2. I will give you an example. Say 1.8 volts and your threshold voltage of the device is 0.8. So 0.80 if you subtract, I get 1 volt right. Now typically if we use a 1.8 volt supply, right your switching thresholds are 0.9, which means that when the voltage, input voltage crosses 0.9 above, it is read as input 1.

As it goes below 0.9, it is read as 0 but if your voltage itself is 1 volt, right then just you have a difference of 0.1 volt, you see. A small noise voltage, some small change in the power supply will switch on from on to off state and that is the reason, it is very critical that you use the whole swing from  $V_{DD}$  to ground right. And that is the reason we use a PMOS at the pull up stage and we use NMOS in the pull down state.

(Refer Slide Time: 18:58)

### Two-Input NAND Gate

- N-input logic implementation requires  $2N$  gates.
- NMOS in series forms an AND logic, while in parallel they form an OR logic.
- Complementary CMOS logics are dual networks (De Morgan's Theorem).

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

❖ Assignment-  
Realize  $F=D+B.(A+C)$  using complementary CMOS design.

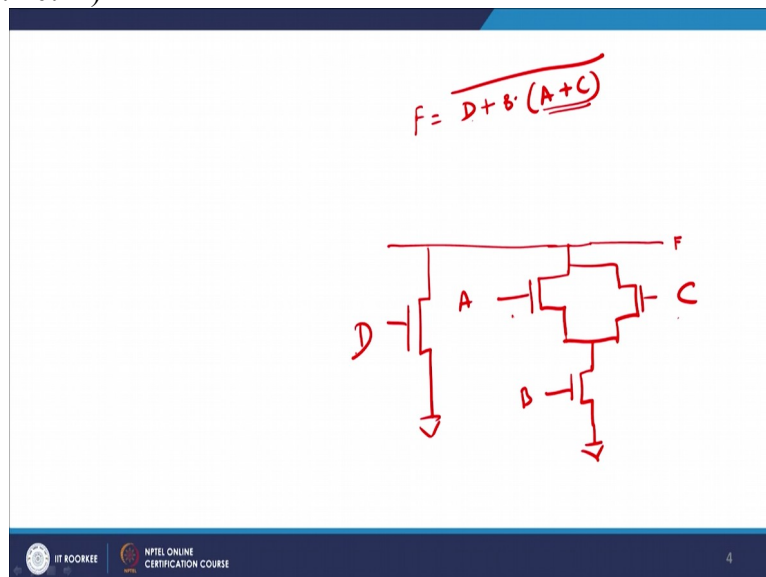
7

Let us look at a two input NAND gate and I suppose you all are aware of it. It is pretty simple. If you look at the logic of NAND, it is something like this. If I have got two inputs, right, let me put it like this right, then this will be, sorry 0 1. Let me do it once again. Let me just rub this off.

So I get 0 0 1 1, I get 0 1 0 1, I get 0 0 will give me 1, 1 0 will give me 1 right and this will give me 0 which means that whenever you get one, these 2 will be on right and therefore this output which is F will go to 0 and therefore output will be 0.

For all other cases whenever you get 1 0, the PMOS is on and even if your one NMOS is on, since they are in series, you have to make them both on together. So either A or B is 0, I will get pull up network to be on and my output will be latched to  $V_{DD}$ . Only in the case, when your 1 1 case, I will get output equals to 0 because both your NMOSs, A and B are switched on simultaneously, fine? And that is what is written all these are written here in this nature.

(Refer Slide Time: 20:42)



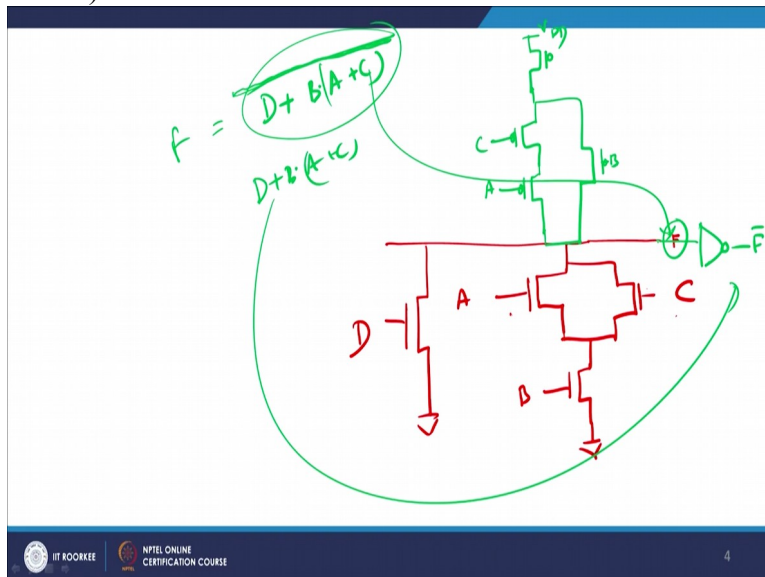
Now let us see, how we can actually make for example, realize, say for example this. The methodology how you realize our combinational logical block in a CMOS circuitry right? Let me say it is D. So it is basically your D. So F equals to D, right? It is equal to D plus B dot A plus C complementary, right? So I have to design this one. So what you do is that, you first of all design, let us say pull down. So pull down, so to do the pull down, I make F here which is the output and then I go pull down.

See, A plus C basically means that you should have 2 gates in parallel to each other, because this is a OR combination. So I should have 2 gates in OR combination and I get something like this, right. So they are OR condition here. So this is your A and this is your C because either of them

is 1, the circuit works right? If this is 1, this is 1, there will be a path available to me. Now D is in series to because this is an ANDing here.

So I will have a B here, right? And then D is ORing with this whole thing. So D is ORing primarily means that, I stop here and then D is ORing means I will have D something like this. D is OR here. Fine. So I have A plus C dot B OR D, right. Now if I have to use it say for example, now this is a pull down network right. This a pull down. Let me design for you the pull up network. Please understand, since this is a complementary logic right, since this is a complementary logic, I would expect to see that the PMOS should be complementary of NMOS which means that wherever you get OR gate, you do AND gate.

(Refer Slide Time: 22:10)



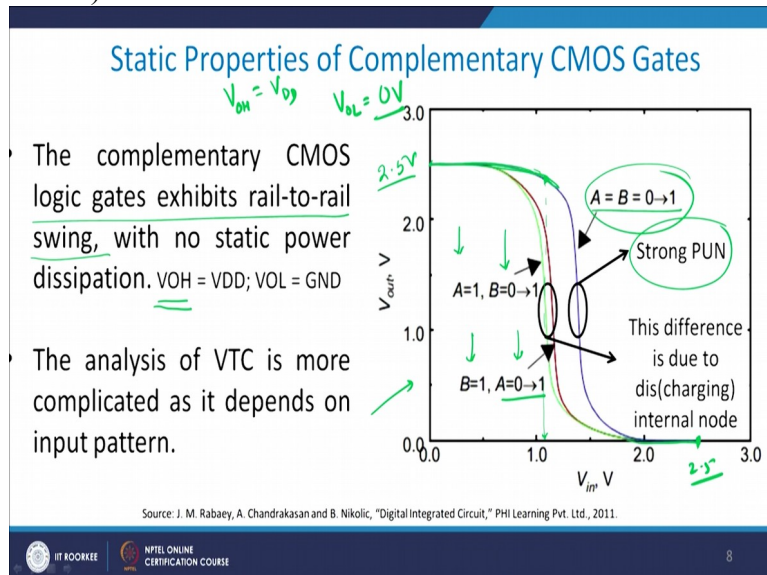
So you just start from here and then make one NMOS here which is A and then make C also in series to this point, right and then since B was in series to A and C, you make something like this. You will make it B like this. B and D was in parallel, so D should be in series again here. So I will do like this. I am sorry. And all will be PMOSs, so all will be PMOSs. So I can have something like this. So this is  $V_{DD}$  and this is ground.

So this is how you realize this D plus B dot A plus C. And why do you get a complementary automatically? Because this is a CMOS structure, right? So if you want to get back the value of F, let us suppose F would not have, did not have this complementary signal, then you need to put a static inverter here and this will give you F bar and therefore, since this is F, you will get F bar

and therefore D plus B dot A plus C will be available to you at this particular point and this will be available at this particular point.

So what we have discussed is, we therefore know that by using certain techniques, I would be able to generate a combinational logical block, right. And this combinational logical block will have these networks available to me in which I can place these networks right and I can have these, such type of designs. So I will recommend that you take any of the standard Boolean expression and try to realize using CMOS switch logic here. For example, this one which we have already done in the class.

(Refer Slide Time: 23:55)



Now look at the static property of CMOS inverter. Let us suppose, as I discussed with you, CMOS inverter logic gives you a rail to rail swing. Rail to rail means  $V_{DD}$  to 0. So the 1<sup>st</sup>, the power rail is  $V_{DD}$  and you have a ground rail. When I say rail to rail swing, I mean to say the output goes from  $V_{DD}$  to ground, right? And this is one of the important merits of static CMOS logic design that you are able to put your output swing up to  $V_{DD}$  and down to ground, the whole limits you can do.

So therefore, my output high,  $V_{OH}$  means  $V_{OH}$  is output high and  $V_{OL}$  is output low. OH is output high and this is equals to  $V_{DD}$ . The output low is basically equals to 0 volts in this case. Now let us see, that you did have but there is a problem here and the problem is, I will just point out from

this figure which you see in front of you, this one. Let us suppose, A and B for example you had a maybe a 2 input NAND gate right and I had A and B both equals to 0 and both goes to 1.

So A and B were both equals to 0 and let us suppose A, sorry A and B were both equals to 0 and both goes to 1. So A goes to also 1, and B also goes to 1. Now when both were equals to 0, please understand, in a 2 input NAND gate, if we look back, the two input NAND gate which was this one, your two PMOS<sub>s</sub> were in parallel, right? Let me erase it.

(Refer time slide 25:34)

### Two-Input NAND Gate (A/2)

- N-input logic implementation requires 2N gates.
- NMOS in series forms an AND logic, while in parallel they form an OR logic.
- Complementary CMOS logics are dual networks (De Morgan's Theorem).

❖ Assignment- \_\_\_\_\_

Realize  $F = D + B \cdot (A + C)$  using complementary CMOS design.

These two PMOS<sub>s</sub> are in parallel which means that if both are 0, both are on and therefore if each carries a resistance of R, then the actual resistance is actually equals to R/2. So the resistance between F and V<sub>DD</sub> rather than being R is now R/2 which means that I can pull up the voltage here to V<sub>DD</sub> in a much easier manner. Agreed? So with this statement or with this knowledge, see what happens.

It means that when both A and B are equals to 0, my output was equals to 1, right. But you see, the output remains 1 for a larger value of your V<sub>in</sub>. You see. From here, it starts to fall down, somewhere here, around 1 volt. You are using a V<sub>DD</sub> of let us suppose, say 2 volt, so around beyond even 1 volt you are able to sustain the input voltage to be equals to, the output voltage to be equals to V<sub>DD</sub> and therefore this is known as a strong pull up network. Why?

Because both your PMOSs are in parallel and therefore the effective resistance goes down and you are able to pull the output voltage at F to  $V_{DD}$  in a much, much better manner. And therefore you see that output remains at  $V_{DD}$  for a larger duration of time. So I am using a  $V_{DD}$  of 2.5 volts. So this is approximately, it goes up to 2.5 here. So both sides, 2.5, right, so your pull down networks are relatively weak in because they are in series.

So I get R and R. So I get 2R as the effective resistance seen downwards and therefore it is very difficult to pull it down and therefore you see, the 0 remains only till 0.5 volts, where as 1 remains till more than even 1 volt, fine and that is the reason you say it is a very strong pull up network. Similarly but you see here very interestingly that for the condition when A equals to 1 and B goes to 0 to 1 and B equals to 1, A goes to 0 to 1, in both the cases, the structure looks approximately the same right? A 1, B goes to 0 to 1 implies that your pull down network gets activated whenever B goes from 0 to 1 and A goes to 0 to 1 implies that it gets activated whenever A goes from 0 to 1, right.

Please understand a two input NAND gate, either of the inputs should be 0 for pull up network to be on, right. Whereas for the pull down network to be on, both the inputs should be equals to 1. So you see, when both the inputs are 0, your output resistance offered is R by 2 whereas if it is only one 0, then you still have a pull up path but the resistance is R only. So it is difficult to pull you up, pull the voltage up from F towards  $V_{DD}$  and that is the reason I wanted to show here.

So it is a bit complicated in the sense that you are not, your output voltages, though it gives you a rail to rail swing but it is not input dependent right and how you, how the input transition takes place, the output depends upon that value as such, right. Let me come to the concept of propagation delay in a CMOS inverter and then we can switch our grounds.



(Refer Slide Time: 29:01)

### Propagation Delay of Complementary CMOS Gates

- Each transistor will replace by its equivalent resistance and capacitance.
- We calculate simple RC delay as-  
( $0.69 \times (R_p \text{ or } R_n) \times C_L$ .)
- The propagation delay depends on input pattern.

Source: J. M. Rabaey, A. Chandrakasan and B. Nikolic, "Digital Integrated Circuit," PHI Learning Pvt. Ltd., 2011.

Now what we do is that, we replace each transistor by a switch and a resistance. So whenever my device is in the on state, this will, whenever my A equals to 0, this will go down. So therefore I write A complement and B complement. When I say A complement and B complement, I mean to say 0 means the switch is on and 1 means the switch is off. Whereas for A and B, when A is equals to 1, it switches on. A equals to 0, it switches off.

So I have a load capacitance here, I have a load capacitance here. Now this is the intermediate capacitance here and is known as the load capacitance. Now we can calculate a simple RC delay from your 1<sup>st</sup> order because these are all first-order circuits. So from your network theory basic course, you can calculate the simple RC delay to be equals to  $0.69 R_p$  into  $C_L$ . why? Because this is a  $C_L$  and this is a net  $R_p$  when you design your external network.

Similarly in the pull down network, it will be  $R_n$  into  $C_L$ .  $R_n$  is the resistance looking downwards, right. Now of course, I will let you know why the propagation delay depends upon the input pattern right. So this is my input, right and this is my output. So when input goes from 1 to 0, the output goes from 0 to 1. So this is 1 to 0 here and this is from 0 to 1, right. So I mean complementary in nature.

Now when both A and B were equals to 1 and both goes to 0, you see the time taken to go to the higher node is much faster and you can understand why. See when both were equals to 1, both these transistors here and here were on, right. And you had these two resistances is coming into



series. So I get  $2R_n$  effectively value. Now what has happened is suddenly, you have made this go to 0, sorry, this 0 and this is also becoming 0. When these two become 0, this switches on and therefore both the transistors help you to pull up the voltage at this particular point to  $V_{DD}$ .

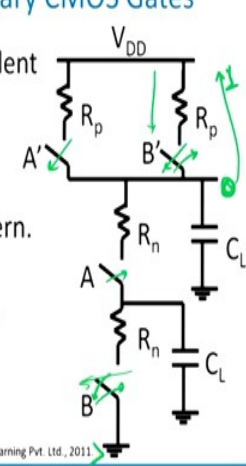
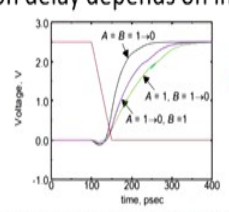
And since the effective resistance  $R_p$  will be  $R_p$  by 2, I will, so this at this point the delay will be half almost and therefore you are able to switch on the value of voltage very fast to one value. This is quite interesting that therefore, your output for a static logic depends upon the input transition also, right. Similarly, if you have, I will not go into details of the 2<sup>nd</sup> one. I leave it as an exercise to you.

(Refer time slide 31:48)

### Propagation Delay of Complementary CMOS Gates

- Each transistor will replace by its equivalent resistance and capacitance.
- We calculate simple RC delay as-  

$$0.69 \times (R_p \text{ or } R_n) \times C_L$$
- The propagation delay depends on input pattern.

Source: J. M. Rabaey, A. Chandrakasan and B. Nikolic, "Digital Integrated Circuit," PHI Learning Pvt. Ltd., 2011

For example, when A was equals to 1 and B goes from 1 to 0 right, when A and B both were equals to 1, your pull down was on, if this was 0. Now what has happened, I am keeping A equals to 1 but I am moving B equals to 0 which means that A equals to 1 means this is on, this is off now and when this is, so let us look at this particular point that let us look at A equals to 1 and B equals to 1 to 0. So when A and B both were equals to 1 which is this one in the on state, so this was on, this was on, this was off, this was off and you had output equals to 0.

Now what has happened? B is equals to 1. B is equal to 1 means B goes from 1 to 0. 1 to 0 means this opens and this closes, right? And therefore, only one path is available you to pull down. There is no pull down path because you have switched off this. So the circuit is broken

down in the pull down. But your pull up path, only one resistance is switched on and therefore this goes from 0 to 1 right. This, the violet line which you see is basically, sorry the green line which you see is basically 0 to 1.

Similarly, when 1 to 0, you see the time taken is relatively smaller as compared to the previous one. I will leave it as an exercise for you to find out. You can get all these discussions in this book by Nikolic, Chandrakasan and Rabaey, one of the standard books which we use across networks, across the whole digital, CMOS digital logic design. You can get the results from this one.

(Refer Slide Time: 32:52)

The slide is titled "Problems of Complementary CMOS Gates" and lists five points:

1. The number of transistor required to implement an N fan-in gates is  $2N$ .
2. The large number ( $2N$ ) of transistors increases the overall capacitance of the gate.
3. Propagation delay of the gates deteriorate as a function of fan-in.
4. The series connection of the transistor in either PUN or PDN network causes an additional slowdown.
5. Therefore, the delay becomes a quadratic function of the fan-in.

The slide footer includes the IIT ROORKEE logo, the text "NPTEL ONLINE CERTIFICATION COURSE", and the number "10".

Let us look at the problems of complementary CMOS. Therefore the problem with complementary CMOS logic is that for N input, you require at least  $2N$  Gates. That we have already seen. For example, two input NAND gate, I require two PMOS<sub>s</sub> and two NMOS<sub>s</sub>. For 3 input NAND gate, I will require three PMOS<sub>s</sub> and three NMOS<sub>s</sub>. So for an N input NAND gate, NOR gate, I require  $2N$  number of gates and therefore the area is relatively large.

Larger number of gate also implies that the overall capacitance is also large because when you have larger number of gates connected to the output, the output sees a larger capacitance which is there with you and therefore, time taken to charge or discharge the capacitance also rises and therefore delay becomes larger. And therefore if you go from a NAND2 to NAND3 logic, your delays will be typically larger in that case, right.

Propagation delay of the gate deteriorates, that is what I was saying. The propagation delay of the gates deteriorates as a function of fan-in. So as the fan-in starts to rise, means you, as large number of gates becomes available to you, you end up having a larger propagation delay and you can understand why? Because your loading, load capacitance which is in the output side starts to become higher and higher.

As I discussed with you, since pull up and pull down networks are complementary with respect to each other, we have to ensure that so if let us suppose your pull up network is in parallel, your pull down network will be in series. If the pull down network is in series, then the overall resistance will be the sum of individual resistances, right?  $R_1$  plus  $R_2$  plus  $R_3$ . As a result, the overall propagation delay will be much, much higher because the resistance is higher.

So you are paying both in terms of higher resistance pull down network as well as higher load capacitive load. And therefore as you can see, the delay becomes a quadratic function of fan-in. And you can understand why assuming quadratic? Because it is once you are paying for resistance, another one you are paying for capacitance and that is the reason, these are the problems of CMOS logic, right.

(Refer Slide Time: 34:52)

The slide is titled "Design Techniques of Large Fan-in" and contains the following content:

- 1. Transistor Sizing
  - To reduce the delay and resistance of the device, the designer must have to increase the sizes.
  - However, increase in size increases the parasitic capacitors which adds its effects in the preceding gate.
  - If the load capacitor is dominated over the intrinsic capacitor then widening the device only creates a self loading effect.
  - Sizing is only effective when the load is dominated by the fan-out.

Handwritten notes in green ink include:  $\uparrow W/L$  and  $R \downarrow$  in the top right corner; a circle around "intrinsic capacitor" in the third bullet point; and a line under "self loading effect" in the third bullet point. A blue arrow points from the underlined text to the "self loading effect" text.

Logos for IIT ROORKEE and NTEL ONLINE CERTIFICATION COURSE are visible at the bottom left, and the number 11 is at the bottom right.

Let me therefore, as I discussed with you, so let us see what are the design techniques available to you for large fan-in circuitry, right? What are the design techniques generally available to you for large fan-in. One is that simply increase the size of the transistor. So if you increase the size

of the transistor,  $W$  by  $L$  increases. You increase the value of  $W$ , right the area goes on increasing and therefore the resistance starts to fall down and you automatically have a lower delay.

However, please understand when you increase the value of your  $W$ , you also end up paying the price of increase in the parasitic capacitances, right. So though you reduce the resistance of the gate, single gate but you add up to the parasitic capacitances of the previous gate and that makes it a bit, slightly difficult to design. That is what I was saying that if the load capacitor is dominated over intrinsic capacitor then the widening of the device only creates a self loading effect.

See, generally a device has got a self capacitance. For example, a  $C_{\text{oxide}}$ , oxide capacitance of the device right. It depends only on the area of the device. So  $\epsilon A$  by  $t_{\text{oxide}}$ . So if we increase the  $W$ , area increases, oxide capacitance increases. That is known as intrinsic capacitance but you also have load capacitances, right. Load capacitances by virtue of a  $C_{\text{GD}}$ ,  $C_{\text{GS}}$  which is basically the overlap capacitances and so on and so forth.

Now if your those capacitances are higher, load capacitances are higher, then increasing the size, if you want to increase the size, you are actually increasing the value of  $C_L$  rather than increasing the value of intrinsic capacitances. And that makes the life difficult as such. And therefore sizing is only effective when the load is dominated by fan-out. So which means that if you are driving a larger number of such devices which is higher fan-out, then only sizing is an important issue or you need to look into sizing issue but if your fan-in is very large right, the sizing will not help you too much as far as designing is concerned for lower propagation delay.

(Refer Slide Time: 37:01)

## 2. Progressive Transistor Sizing

- This technique reduces the dominant resistance while keeping the capacitance in bounds.
- Progressive scaling of transistor is beneficial:  $[M_1 > M_2 > M_3 > M_4]$
- The progressive scaling is easy in schematic diagram but it is not as simple in layout.

IT BOONKEE  
NPTEL ONLINE  
CERTIFICATION COURSE

12

Now, let me give you a concept of progressive transistor sizing. See, if you look carefully at the pull down network which is shown here in this case, so when both in 1, 2, 3 and 4, let me say this is  $In_N$ , there are  $N$  number of logics. So there are  $M_1, M_2, M_3, M_N$ . So there are  $N$  number of gates available here. Now suppose all the gates are having 1 1 1 1 1.

All the NMOSs are on, then you can see that if you look at the delay, I get a 0.69 right multiplied by  $M_1$  which is the this one multiplied by  $C_1$ , resistance of  $M_1$  multiplied by  $C_1$  plus resistance of  $M_1$  plus resistance of  $M_2$  into  $C_2$ , right plus resistance of  $M_1$  plus resistance of  $M_2$  plus resistance of  $M_3$  into  $C_3$ . You are getting my point? This plus this plus this, this plus this plus this multiplied by this.

This plus this multiplied by this and this multiplied by this, this is also known as Elmore delay. So you see, this  $RM_1$  is coming thrice and if there are  $N$  number of gates, you will get  $RM_1$   $N$  number of times. So, it is always advisable to make the lowest gate right? The lowest gate means this, you got it in series, the largest gate. You make it largest, this resistance will be the smallest and therefore you go on adding it, you get the smallest delay with respect to the overall delay.

So it is always advisable that try to keep the aspect ratio of the gate which is farthest away from the input largest. Once you ensure that, your overall delay starts to reduce and this is what is known as progressive scaling in transistor. So  $M_1$  is greater than  $M_2$  is greater than  $M_3$  is greater

than  $M_4$ , right. Though it looks very simple on a pen and paper, in actual layout it becomes very difficult to achieve it.

(Refer Slide Time: 39:18)

### 3. Input Reordering

- All the signals in the complex logic blocks might not appear at the same time due to propagation delay or preceding logic gates.
- The signal, last to all the inputs which have a stable value can be called as a critical signal on the path over which the ultimate speed of the structure can be calculated is called critical path.
- Putting the critical path transistor closer to the output gives a higher speed of operation.

13

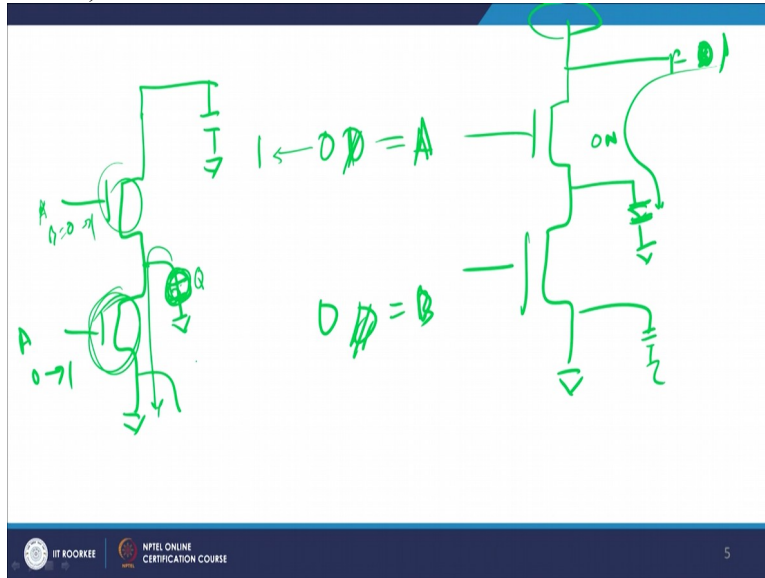
What is known as input reordering? I will just use a single term here. You can go into it later on. Now try to make, see whenever you have a combinational logic, you will have  $N$  number of signals coming together, right? For example, I will just you a small example, let us suppose I have got these 2 signals here right and I have got this  $F$  here and this is my pull down network. So there are 2. I have got  $A$  and  $B$  here. Now let us suppose  $A$  arrives earlier.

So even if  $A$  arrives earlier, this switches on but still, suppose  $A$  equals to 1 and this switches on, my  $B$  is equal to still 0, right and it will go to 1 but it will go to 1 at a later stage. But till it goes to 1, my  $F$  cannot go to 0 because there is no direct path between  $F$  and ground. Are you able to get the picture? Which means that, you have to wait till the  $B$  signal comes into picture, right. So we define  $B$  signal to be as the critical signal.

So what is the critical signal? Critical signals are those signals which are the last to arrive and which are, when they come last to arrive, they evaluate the output also last. So they help you to, so you have to wait till the critical signal arrival for the output to go to 0 or 1, right? The rule of thumb is and there is a reason for that that you try to keep this, if  $B$  is the critical signal, try to keep it closest to the output. So putting the critical path transistor closer to the output, gives a higher speed of operation.

This is a rule and there is a reason for that. You please find out yourself what the reason is, but this is typically the reason that the critical path signal should be kept closer to the output. It has to do with the discharging, output discharging, right. I will tell you how, I can give you a reason for that.

(Refer Slide Time: 41:08)



Let us suppose I have this as the consideration and this is A and this is B, right? This is B and this is, so this is your F and this is your pull up network, something is here. Let us suppose B is the critical signal, right? A is the non-critical signal. So when A becomes equal to, both were 0 and output was equals to, so both sorry, I am sorry. Both were equals to let us suppose 0, output was equals to 1 because both were off. Now A goes from 0 to 1 but B still remains 0 and it has to wait till then.

So when it goes to 1, this becomes on. This becomes on means that, so you have a capacitance here and a capacitance here also. Some amount of this 1 will appear across this capacitance here but it has to, some amount of charge at F will appear will charge this  $C_L$ . But then, it has to wait till B arrives right? So what you try to do is quite interesting is, that you try to make your B arrive here and A arrive here. So A has already gone from 0 to 1 right and therefore, so this is grounded, this is there and this is with me and I have a capacitance loading here.

So A goes from 0 to 1 implies that this has switched on, all my charge, extra charge available at this point has gone to the ground and therefore, simply if this goes from, B goes from 0 to 1 now,



this becomes on, then this extra charge which was already available here and need not be discharged. It has already been discharged within the time. So you do some amount of time sharing. So the time till which your critical signal was not appearing to you, you discharge the extra capacitance or discharge the extra charge at the drain end of my transistor A. And therefore, you need not therefore discharge large charge. You have to discharge a lower charge and therefore your speed becomes larger, right.

(Refer Slide Time: 43:08)

#### 4. Logic Restructuring

- Manipulating the logic equation can reduce the fan-in requirement and thus reduces the gate delay.

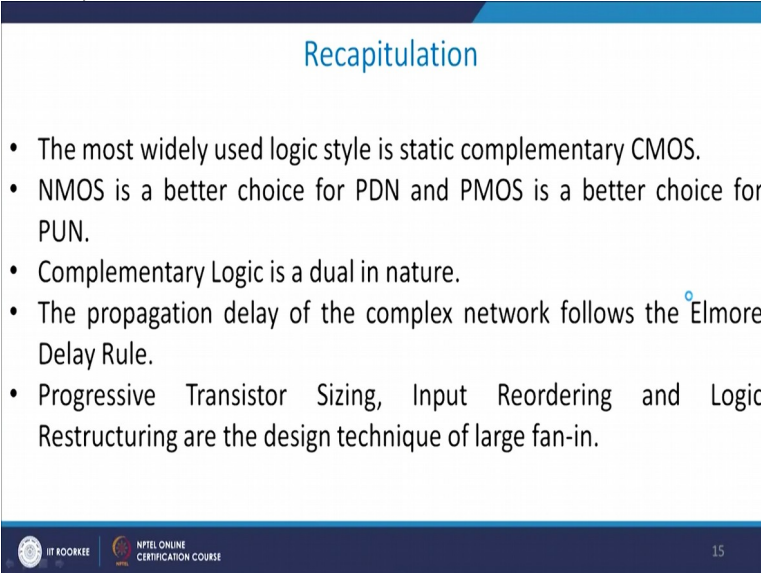
The diagram shows a logic restructuring process. On the left, a 6-input NOR gate is connected to an inverter. An arrow points to the right, where the same logic is implemented using two 3-input NOR gates and one 2-input NAND gate. The 6-input NOR gate is crossed out with a green line. Handwritten green annotations include 'f' on the 6-input NOR gate, 't<sub>ms</sub>' near the 2-input NAND gate, and 'f' and 't<sub>ms</sub>' near the 3-input NOR gates.

Again one methodology is that logic restructuring methodology which is logic number 4. You have to manipulate it in such a manner that you try to make it as symmetrical as possible. So when you make it symmetrical, the delay, so typically let us suppose it was not symmetrical and let us suppose, this reaches 1 and this reaches 0 right, so output will be 0 right because it is an AND gate. But if this is 1 and if this is let us suppose, reaches some delay by say  $t_{ms}$ , then this output has to wait till  $t_{ms}$  for the output to appear.

So what you try to do is, you try to make it symmetrical in nature. So, for example, 1, 2, 3, 4, 5, 6. 6 input NOR gate can be broken up into 2-3 input NOR gate and 1 NAND gate. So it is a very high fan-in, not a very good idea to manipulate, break it into 2 smaller fan-in transistors and you try to make it more symmetrical in design. And that is known as logic restructuring, right?



(Refer Slide Time: 44:08)



### Recapitulation

- The most widely used logic style is static complementary CMOS.
- NMOS is a better choice for PDN and PMOS is a better choice for PUN.
- Complementary Logic is a dual in nature.
- The propagation delay of the complex network follows the Elmore Delay Rule.
- Progressive Transistor Sizing, Input Reordering and Logic Restructuring are the design technique of large fan-in.

IIT BOOBKEE NPTEL ONLINE CERTIFICATION COURSE 15

So let me recapitulate what we done till this module. The most widely used static module design is CMOS logic. As we discussed just now, the NMOS transistors are better pull down networks, PMOS devices are very good pull up networks, they are part of pull up networks. Complementary logic is very, is dual in nature. Please understand, this is very, very important when you are designing combinational logic. But whenever you are doing a complementary logic, your pull up network and pull down network are dual of each other which means that if upper 2 transistors are in series, the lower 2 will be in parallel and vice versa.

So you should keep in mind and should do a large amount of practice of, so if I give you a Boolean expression, you should be able to design its logical function or the complementary logic. The propagation delay of the complex logic follows NMOS delay rule which we have discussed and therefore, it is always advisable to keep your transistor which is most away from the output, the largest in size, therefore resistance falls down and that is the one methodology.

The 2nd is that when you do, so this is known as a progressive sizing of the transistor. You also need to understand that the critical signal should be kept very close to the output for your lower delay and then you also have to do logic restructuring. Try to keep your fan-in low and try to make it a symmetrical path so that you do not have glitches or there is no problem in this case right. So these are the few statements or the few areas in which people have worked on. And I

hope you have understood this module in a better manner. Okay. Thanks a lot. Thank you for your patient hearing.