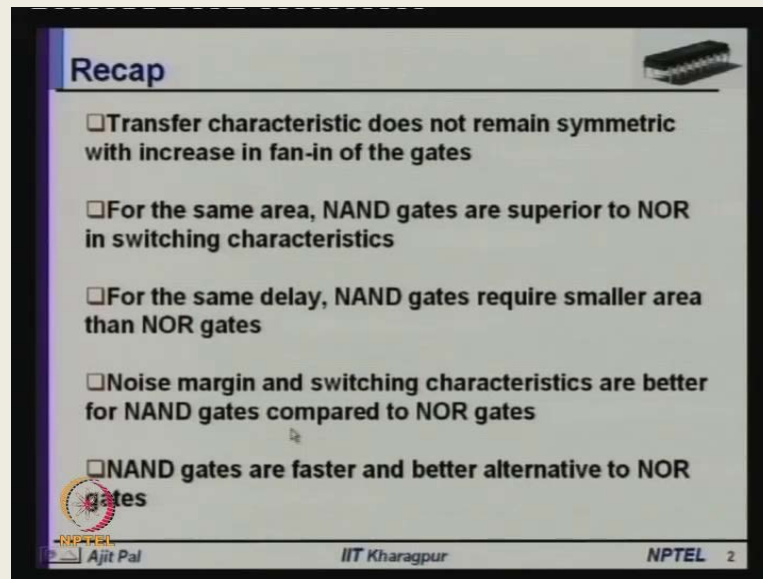


Low Power VLSI Circuits and Systems
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. # 11
Static CMOS Circuits – II

Hello and welcome to today's lecture on Static CMOS Circuits; this is the second lecture on this topic. In the last lecture we have seen, how you can extend the concept of CMOS inverter to realize an NAND gates and NOR gates. And we have studied the characteristics of **these** those NAND and NOR gates.

(Refer Slide Time: 00:47)



The slide is titled "Recap" and features a small image of a CMOS chip in the top right corner. It contains five bullet points, each with a square checkbox:

- Transfer characteristic does not remain symmetric with increase in fan-in of the gates
- For the same area, NAND gates are superior to NOR in switching characteristics
- For the same delay, NAND gates require smaller area than NOR gates
- Noise margin and switching characteristics are better for NAND gates compared to NOR gates
- NAND gates are faster and better alternative to NOR gates

At the bottom of the slide, there is a logo for NPTEL (National Programme on Technology Enhanced Learning) on the left, and the text "Ajit Pal" and "IIT Kharagpur" in the center, and "NPTEL 2" on the right.

And here is the agenda of today's lecture of course, before giving the agenda of today's lecture, here is a brief recap what we discussed in the last lecture. And or rather we can summarize what we discussed in the last lecture.

We have seen, that transfer characteristic does not remain symmetric with increasing fanning of the gates; that means, when you are realizing two input NAND gate, three input NAND gate or two input NOR gate or three input NOR gate. As you are increasing the transfer characteristic becomes asymmetric that means, the switching point of the

gate moves either towards left or towards right. **Second** our second observation was for the same area NAND gates serves superior to nor gates in switching characteristics we have seen that, if we realize gates with minimum area than NAND gate is superior to NOR gates in terms of switching characteristics of course, area remaining same.

Also we have seen for the same delay NAND gates require smaller area than nor gates that means, if the area is not a constraint. If we want the NAND and NOR gate should perform equally in terms of switching characteristics, then with increased area for the NOR gates, we can achieve same performance for both NAND and nor gates. Our fourth observation was noise margin and switching characteristics are better for NAND gates compared to nor gates.

So, because of all these features obviously NAND gates are faster and better alternatives to NOR gates. However whenever you go for realizing real life circuit, you will see that both NAND and NOR gates are used as part of the standard cell library. Because, in whenever you are realizing a complex circuit it is not the delay of individual gate but, the delay of the entire circuit is important.

(Refer Slide Time: 03:02)

Combinational Circuits

- Combinational circuits perform Boolean operations on multiple input variables and produces output as Boolean functions of the input variables
- Generates output based on the input values after a certain time delay and preserves the output as long as power supply is provided

The diagram shows a rectangular block labeled "Combinational Circuit". On the left side, three arrows point into the block, labeled "Primary inputs". On the right side, one arrow points out of the block, labeled "Output". At the top of the block, a line connects to a terminal labeled "Vdd". At the bottom of the block, a line connects to a terminal labeled "Vss".

Ajit Pal IIT Kharagpur NPTEL 3

So, in that context we shall consider a combinational circuits, obviously in inside this circuit we have a complex network of different types of gates. May be combination of NAND, NOR, EX, AND, OR, exclusive OR different types of gates and here are the two important characteristics of combinational circuits. Number 1 is combinational circuits

perform Boolean operations on multiple input variables and produces output as Boolean functions of the input variables.

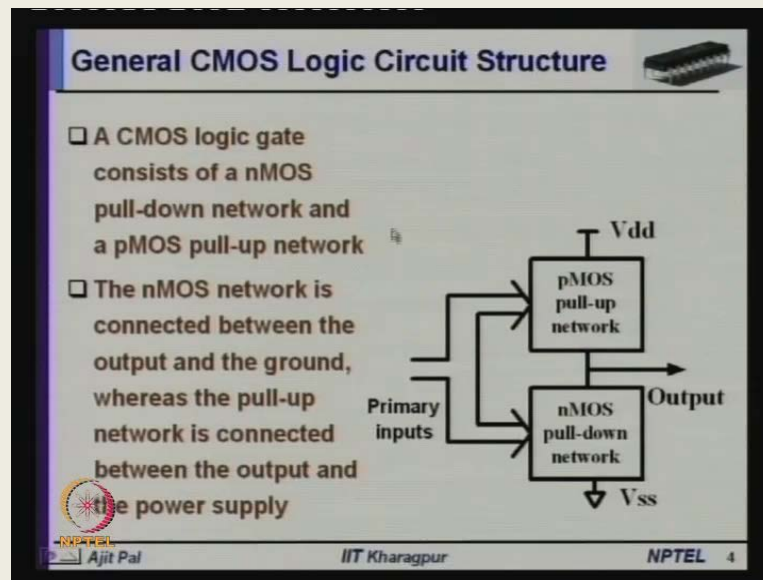
Now, you may be asking what is the number of primary inputs are which are used in present day v l i circuits. You will be surprised to note that, the number of inputs of present day combinational circuits can be as high as 100 or even 150 that is the number of peoples fanning to the combinational circuit is increasing.

And whenever you are realizing a combinational circuit **with such** with such a large fanning obviously it cannot be realized by a single gate you have to realize by a network of gates. Then second characteristic is it generates **outputs** output based on the input values after a certain time delay and preserves the output as long as power supply is provided. That means this is another characteristic of combinational circuits that mean output is dependent on the primary inputs not on past history.

However output value that we get will be obtained after certain delay, because the combinational circuit will have some final delay. As we have seen even a single gate NAND gate or nor gate has **has** some final delay; whenever you are realizing a circuit with a complex network of gates. Obviously delay will be larger maybe 3 or 4 gate delays or more. That means it will be a multi level network gate realization.

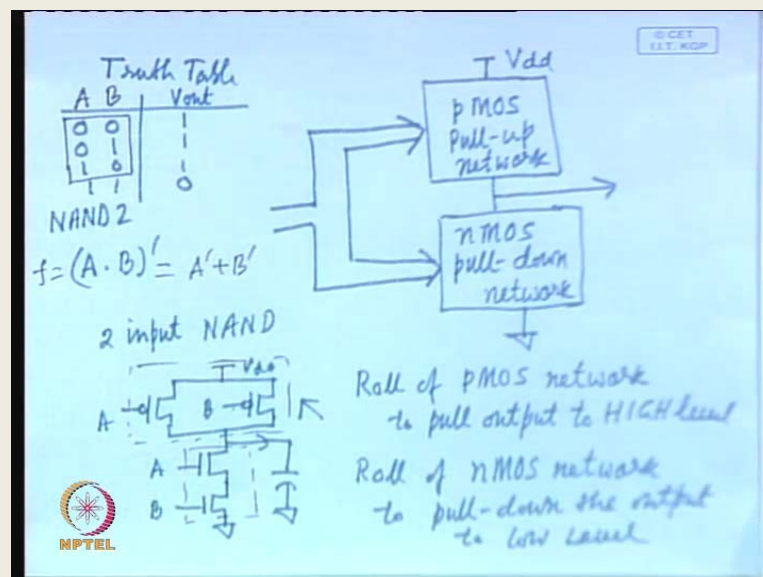
And another point is output remains steady as long as power supply is provided and that is particularly true for static CMOS circuit. Later on we shall see, the there is another version as I mention dynamic CMOS circuits **which is a** which also realizes combinational circuit. But, there we will see that output will change unless it is refreshed at regular interval but, that we shall discuss later on **ok**.

(Refer Slide Time: 05:43)



Now, let us have a look at the general CMOS logic circuits structure we have already seen how we can realize NAND and NOR gates. We have seen that NAND and NOR gates idea multiple NAND and NOR gates concept can be extended to realize any complex CMOS circuit, where the general logic circuit structure is like this let me redraw it again.

(Refer Slide Time: 06:16)



So that you have got a **p MOS** p MOS pull up network and which is connected to the supply voltage V d d. And you have a n MOS pull down network, which is connected to

ground. Output is taken from the junction of the p MOS network and n MOS network and common inputs are applied to both the p MOS and n MOS circuit.

And we have already seen **how it** how you can realize NAND and nor gates for example, two input NAND gate. A two input NAND gate we have seen will require a network of two p MOS transistors and a network of two n MOS transistors in series. So, here you will apply let us assume A B, A B and you will take output from here. So, **this is the** this is your p MOS network that means this part is your p MOS network, which is connected to V_{dd} and this is your n MOS network, which is connected to ground and you are taking output from here.

Now, what is the role of the p MOS network **role of p MOS** network, so what is the role of p MOS network is to as the name suggest it is a pull-up circuits. That means the it pulls up the output capacitors voltage to V_{dd} level, that means the load capacitance which connected to the output, it pulls it up to V_{dd} . In other words the output whenever you are expecting an output 1, then there will be a path through the p MOS network. That means p MOS network will be on, whenever output is supposed to be 1; in fact you can do the logic design based on that idea.

Similarly the, what is the role of n MOS network, so role of p MOS network is to pull the output to high level. And role of n MOS network is to **pull the output**, pull-down the output to low level. So, based on this you can essentially realize a circuit that means when the output is to be 1 this p MOS network will be on there will be path of V_{dd} to the capacitor. Similarly, when the output is to be 0, it will be 0 **for the** for a given input combination the path will be through the n MOS network, this is how we can realize a circuit.

Now, let us take of some examples (Refer Slide Time: 10:00) how can I realize AND and OR gates we have already discussed the realization of NAND gates. Now let us consider the realization of AND gate, as you have seen the realization of NAND and nor gates come very naturally to static CMOS circuits. You have seen you will just realize a series connection of n MOS transistors and a parallel connection of n MOS transistors and realize the circuit.

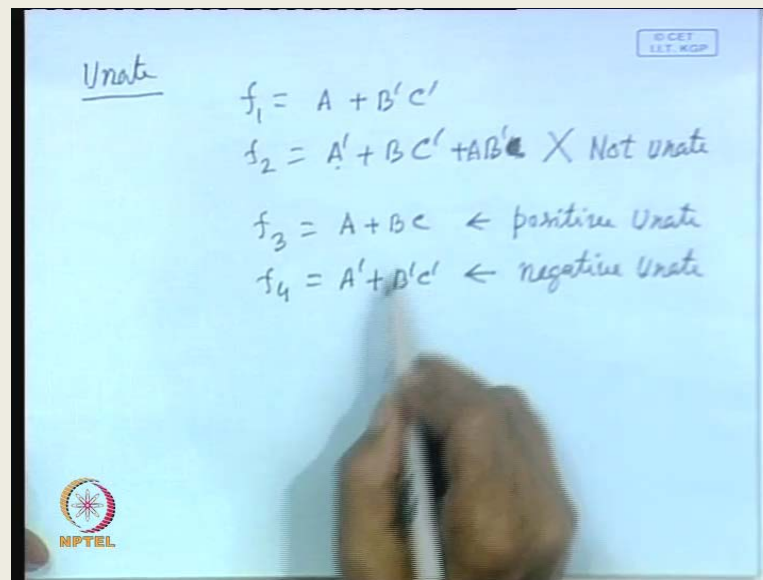
And output function in this particular case that means NAND two the output function is represented by $A \cdot B$. If you consider the truth table truth table will be, we have got two inputs there will be four combinations 0 1 1 0 and 1 1.

So for non NAND gate the output v out will be output 0, whenever input is 1 1 in rest of the cases output is 1. So, we find that output will be 1, whenever you apply any input combinations like 0 0 0 1 and 1 0, that means whenever you apply these inputs a path has to be established for through the p MOS network. How it can be done as you know, when we apply when we apply 0 to a p MOS transistor it transform. So, we find that if anyone of if anyone of the two transistor is apply, I mean if we apply 0 to anyone of the two transistors output will be 1 and in fact that is the reason why we apply.

I mean you can realize based on that, if you simplify this you will get, essentially if we will combine these two, we will get A bar and here we will get B bar. So, essentially you are applying A bar and B bar and you are realizing this part of the network. Similarly, for 1 1 output will be 0, as you can see 1 1, means you will require two transistors in series and whenever both of them are 1 1 output will be 0. And this is the function f that it realizes and if you represent it will be I mean in extended form A plus plus B bar B plus.

One important characteristic of this is you see all the output function in a minimized form has got it is literals that means A and B in complemented form. What kind of function it is in, all of you must have attended a course on switching circuits and logic design. And you have been introduced different types of logic functions.

(Refer Slide Time: 13:28)



You may recall that a Boolean function can be said unate if you may have heard the term unate, what do you really mean by unate, when we call a function unate. Unate is a function is unate, when the literals come either in complemented or in uncomplemented form that means for example, A has A present here in uncomplemented form B and C both are present here in complemented form.

so this is a unate function but, if, **if, if, if** another function you write say f_2 , which where it is $A + Bc' + AB'$ or say $A + Bc'$. Let us assume this is the function in this case as you can see, A is appearing both in complemented and uncomplemented form, B is also appearing in complemented and uncomplemented form, so it is not unate function. On the other hand, if all the variables appear in uncomplemented form say $A + Bc$, where none of the variables has appeared in complemented forms. So, it is called positive unate function **positive unate function**.

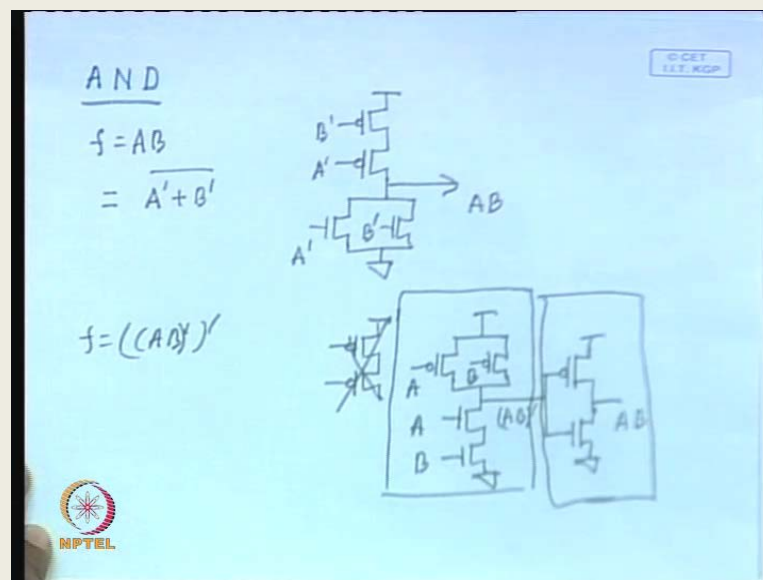
And similarly, f_4 which is $A' + B'c'$, where all the variables have appeared in complemented form it is called negative unate function. And in fact for unate function these expressions are you know there is no alternative, that means these representations are unique. That means minimum sum of product **product** form is unique so they are the canonical representation of the function.

So, you can see here f_3 where all the variables are in non complemented form and here all are complemented form. Now, **if we consider** if we look at this realize the

function of the NAND gate (Refer Slide Time: 15:57), what does it tell both the variables have appeared in complemented form that means, whenever you are realizing a function f.

And you are applying some inputs here it will be a negative function of the inputs, that means a CMOS gate realizes a negative unate function in terms of its input variables. That means all the variables will be in complemented form, we have diverted to this the reason is whenever you are realizing AND gate it is not so.

(Refer Slide Time: 16:40)



Whenever you are realizing and gate f is equal to $A B$, obviously you cannot really realize an **NAND** AND gate without, I mean just by applying A and B of course, you can use De-Morgan's law to represent it different ways. For example, we can have it you can represent it in this way A dash plus B dash bar. Now, you can realize this one, this function can be realized A dash plus B dash bar.

So, now, you can see the functions are in complemented form the **the the** input, that means you can realize an AND gate in this way. Obviously you will realize you will require **two** two p MOS transistors and two n MOS transistors and to the n MOS transistors the inputs will be A dash and B dash here also you will you will apply A dash and B dash. So, here it will realize $A B$ essentially it is realizing this **this** A dash A dash plus B dash bar.

So, here we have assumed that the complemented variables are available as inputs but, if the complemented variables are not available as inputs, how can you realize it. Another alternative realization is by using, what you can do you can realize f is equal to $A B \bar{\bar{}}$ sorry bar bar, that means you will be realizing a an NAND gate and put an inverter. For example, you already know, how you can realize a NAND gate, we input NAND gate sorry, you will realize NAND gate you will require two p MOS transistor in parallel. And two more two n MOS transistors in series.

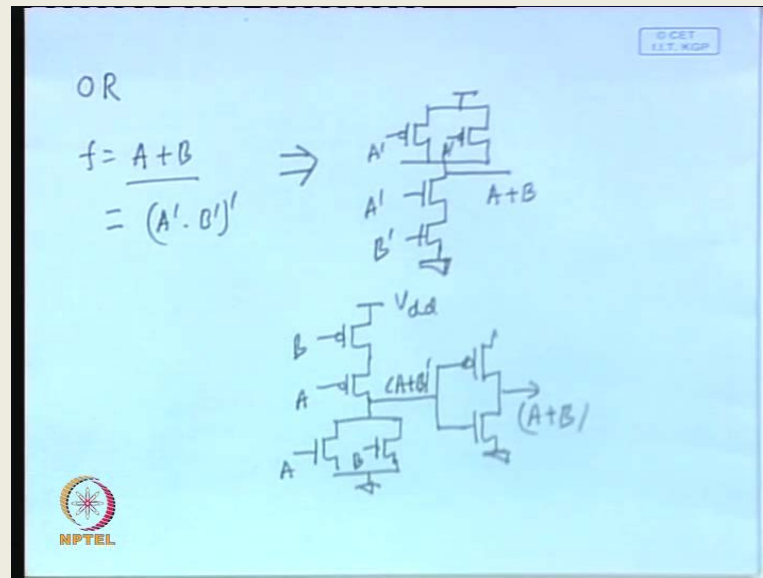
So, here we will apply $A B A B$ and this is connected to ground; then of course, at the output you will again require an inverter. Because, here it realizes $A B \bar{\bar{}}$ so that has to be complemented to get the function of A of an AND gates, so here you will get $A B$.

So, here you see you can realize and gate in two ways, this is how you can realize when the inputs are available in complemented form also in complemented form. On the other hand if the inputs are not available in not available in complemented form, then you will require two levels of realization. Because, in the first level you will realize NAND gate two input NAND gate and you will invert it to get the AND functions.

So, both are possible, however the second type this will require larger delay but, this will require, I mean in the complemented inputs as I, I mean must be available to you if the complemented inputs are not available this is how you can realize. So, both the configurations are used in practical realization.

In a similar way that means what you have done here, so you have expressed the functions in terms of two, you know this is a this is realizing a negative function. This is realizing a negative function, because this whatever is input it gives bar implement bar complement of that. So, you can see essentially you have expressed the function in terms of negative functions you have decomposed the functions in terms of negative functions. In fact there are sometimes back in 70's lot of research were carried out to realize combination CMOS combination circuits by expressing it in terms of negative unate functions anyway, this is how you can realize AND gates.

(Refer Slide Time: 21:13)

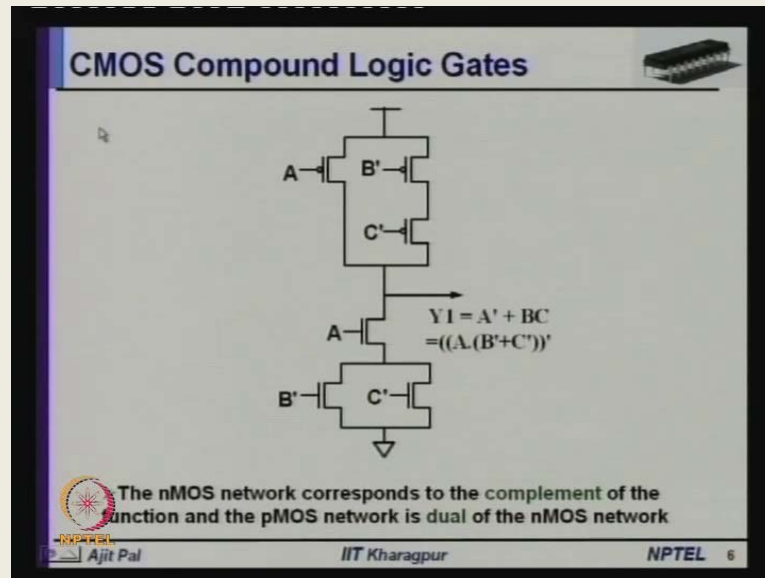


Now, let us consider the realization of OR gate, in a similar way or function is equal to f is equal to A OR B it can be represented as $(A \text{ bar and } B \text{ bar}) \text{ bar}$. So, this is how you can represent it by following the De-Morgan's law and corresponding realization is the you will require an OR gate in this way $A \text{ bar and } B \text{ bar}$. So, you will require and so $A \text{ bar } B \text{ bar}$ and here, it will be just the.

So, we find that in this particular case also, you are applying complemented inputs because, this is a **this is a** positive unit function a positive unit function cannot be directly realized by a CMOS gate. So, we have applied complemented inputs to realize OR function; of course, you can you can realize it in a different way as I have already told, you can realize the NOR gate first. You can realize NOR gate this is a typical NOR gate realization $A \text{ B } A \text{ B}$ and then, put an inverter.

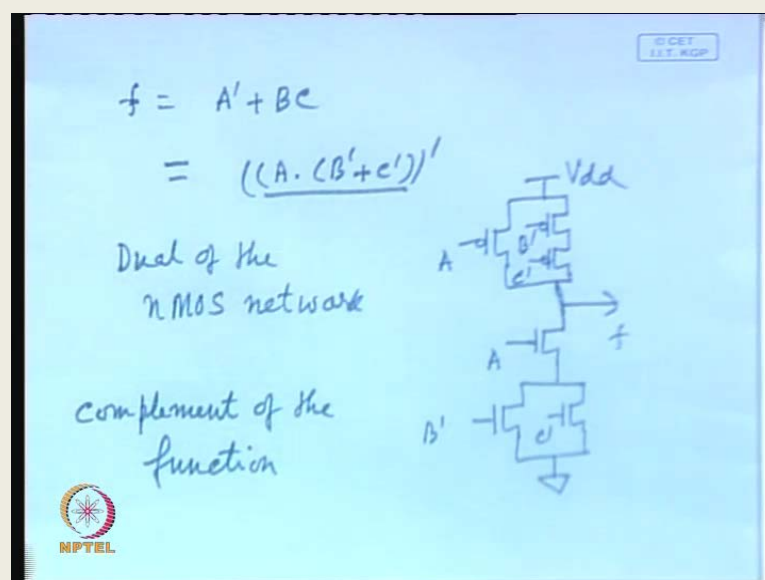
So, this is an alternative realization for OR gate, so that means here you are realizing $(A \text{ or } B) \text{ nor}$ function, then here you are getting or function. So, this is **this is** how you can realize or gate.

(Refer Slide Time: 23:16)



Now from this, what can I say **we can we realize** we can realize AND gate and OR gate and also, some complex functions but, can we realize any unconventional function. So, we have seen in terms of NAND gate, NOR gate, OR gate, AND gate realizations, we have seen. Can we realize any complex function by extending the idea that we have discussed so far.

(Refer Slide Time: 23:53)



So, that can be done we can realize any complex circuit for example, you have to realize a function f is equal to A dash plus B C this is this is not a **this is not a this is not a**

positive unate function or a negative unate function how can you realize this. So, we have seen from this realization (Refer Slide Time: 24:18), what it really does or if we consider, what this part of the circuit does. This part of the circuit realizes the complement **complement** network this is the complement.

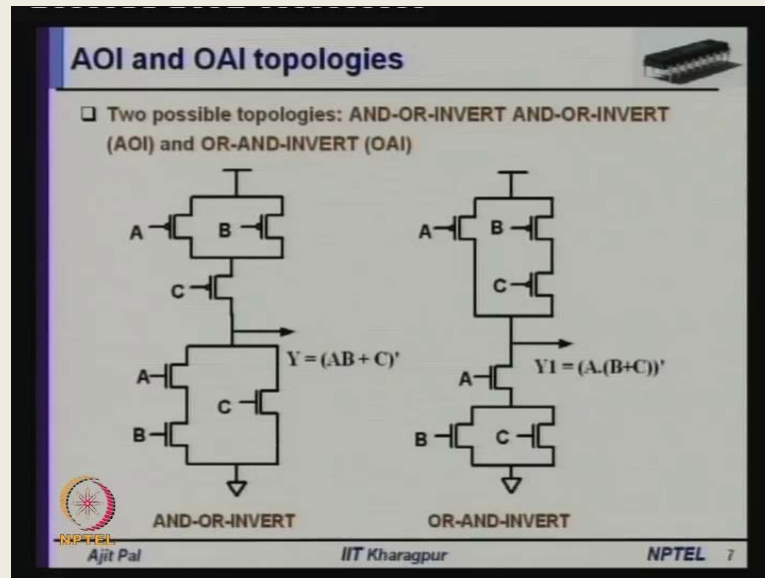
So, $A \text{ dash } A$ or $A \text{ bar}$ plus $B \text{ bar}$, so this is the n MOS network realizes the complement function. And the p MOS network is essentially dual of the function the dual of that network; dual means where whenever it is series it is parallel, whenever it is parallel it is series. So, that is the concept that you can use to realize any complex functions like this.

So, this can be realized in this way this can be expressed in this form, $((A \text{ dot } (B \text{ dot } \text{ or } C \text{ dot})))$, then you can complement it. So, complement is $A \text{ and } B \text{ dash plus } C \text{ dash}$ and then complement of this is how **this is how** it can be represented. So, what is the corresponding realization this is the network n MOS network. So, n MOS network will be the series of A and you will apply $B \text{ bar } C \text{ bar}$ in parallel and connect it to ground so this will be your n MOS network. So, the p MOS network, so it is the **complement of the** complement of the function.

So, and the p MOS network will be dual of this network, so there it will be, this is in series, so A will be applied parallel, here we will apply A and these two here they are in parallel they will be in series. So, you will connect it to V_{dd} and you will take output from here, so this realizes f, that means here the p MOS network is dual of the n MOS network; you will apply the same input $A \text{ B dash } C \text{ dash}$. So they are in series and then you will get the function, so we find that this is how you can realize any complex function.

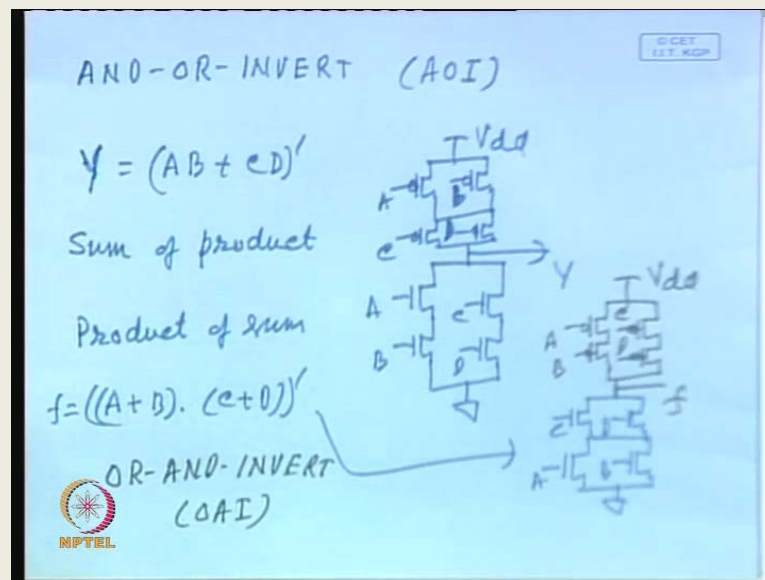
So, we have extended the basic idea of realizing simple gates to realize complex gate, you will be asking why we should go for realizing complex gates. The reason for that is there are **basic** two basic approaches, one is you can use standard cells to realize any given Boolean function. So, in that case, the delay can be quite long, on the other hand if you realize a complex CMOS circuit but, it should not be very complex, that means the number of transistors in series and parallel should not be very large. Maybe upto four transistors in series and four four transistors in parallel will be allowed beyond that delay will be too long. So, as long as the number of inputs remain within three or four and function is complex, you can realize any Boolean function used in this approach.

(Refer Slide Time: 28:08)



Let us, see how it can be done, essentially there are two basic approaches or two basic topologies, one is known as AND-OR INVERT.

(Refer Slide Time: 28:31)



One approach is known as AND-OR-INVERT in short AOI, this is one approach that means you can express the function in terms of AND-OR-INVERT. For example, say you are realize a function Y is equal to A B plus C. So, this is a AND you could have puts the D also here then, so AND of two product terms all of them and then, complement so AND-OR-INVERT.

So, this **topo** this particular function can be realized **by using the by using by can** be can realize it you can implement it by following the **the the** corresponding topology will be say A B in A B will be in series, then C D will be also in series and they are in parallel. So, this will be your **p** n MOS network using AND-OR-INVERT approach and corresponding p MOS network **will be** will be just the complement of it. Complement means A and B will now be in parallel, so A and B will be in parallel and C and D also will be in parallel.

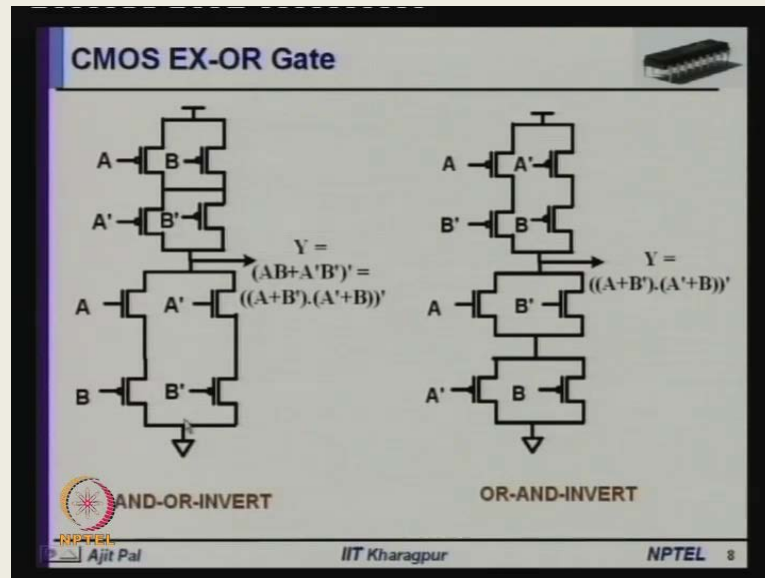
So, this is D this is C and we will take the output from here, this will be connected to V d d and this will be connected out, so this line realizes is Y. So, following AND-OR-INVERT approach you can realize a function in this manner. Similarly, you can also use another approach depending on the representation of the function, we know there are two possible minimal form. One is known as sum of product **sum of product** this is essentially, complement of the sum of product. Another is product of sum **product of sum** that means it can be like this say **a** (A plus B) into (C plus D). So, here you have taken the product of sum, that means and if the function is like this and complement of this **this** can be realized very easily.

For example, there the corresponding topology this function will be realized in this way. So, you will be having A and B in parallel **A and B in parallel** A B connected to ground and C and D again in parallel **C and D in parallel** and this will form the n MOS network. And p MOS network will be the dual of that that means A and B will be in series A and B will be in series and C and D also will be in series and they will be in parallel.

So, there is C and D and here is your V d d, so this suppose this is your f, so this realizes it. So, this is the corresponding OR-AND-INVERT topology, that means **you can realize** you can use one of the twos topologies for realizing a Boolean function. You can either represent in sum of product form or product of sum form to realize the complex CMOS function using one of the two topologies AND-OR-INVERT or AND-OR-INVERT, so this is in short (O A I) OR-AND-INVERT.

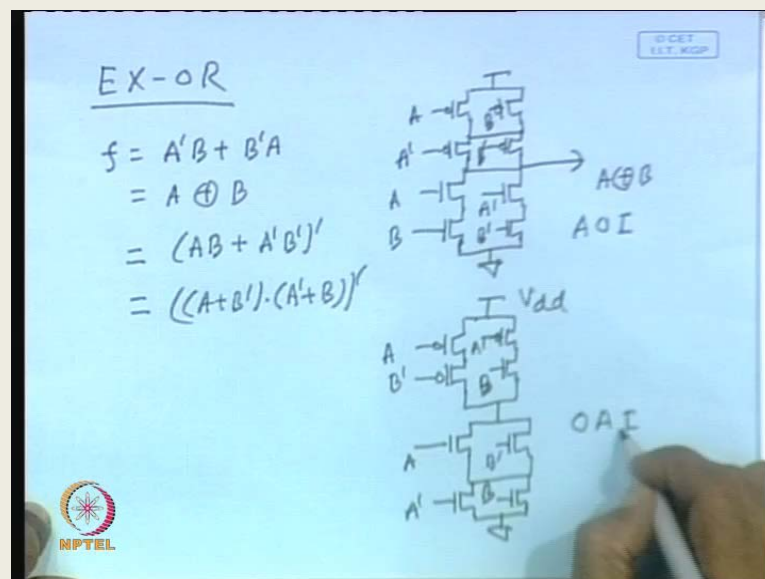
But, in all the cases as I told the output is a negative function, realized as a negative function respective of the way you are realizing the circuit. Maybe in AND-OR-INVERT topology or by using OR-AND-INVERT topology the function is negative in terms of the inputs.

(Refer Slide Time: 33:29)



Let us take up an example complex CMOS EX-OR gate, so here the realization of a CMOS EX-OR gate is shown. Here, you can see EX-OR function has been realized in AND OR INVERT form, AND OR INVERT means $A B A B$ or $A \text{ bar } B \text{ bar}$ sorry this is I mistake this will be **this will be** this will be let me draw it here. There is some mistake there these are n MOS by mistake it has become CMOS.

(Refer Slide Time: 34:05)



Say EX-OR gate, **EX-OR** you are realizing EX-OR function, now EX-OR function as you know f is equal to $A \text{ bar } B$ plus $B \text{ bar } A$ or you know it is also represented as

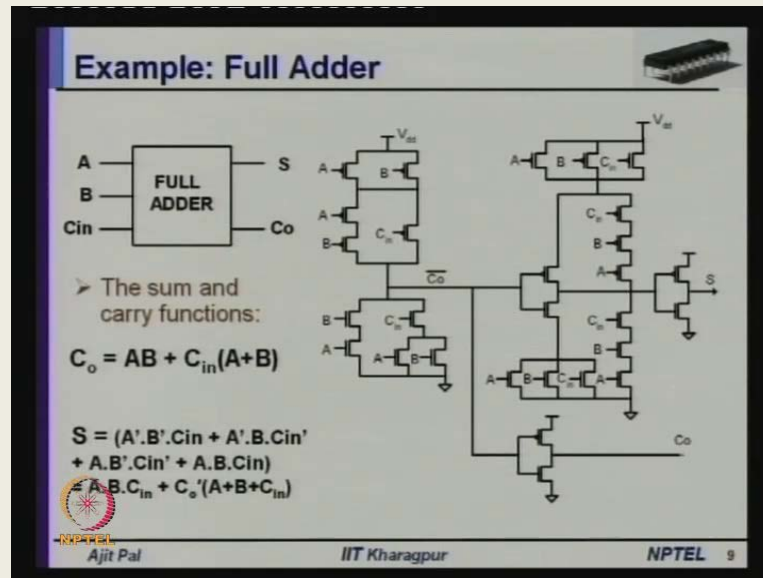
A exclusive or B. Now, you can realize it in this manner you can by using the Boolean algebra you can express it in this form say $(A B \text{ plus } A \text{ dash } B \text{ dash}) \text{ bar}$. So, either it can be represented in this way or it can be represented as $((A \text{ plus } B \text{ dash}) \text{ into } (A \text{ dash or } B))$ and then complement of it.

So, depending on that you can have AND-OR-INVERT or OR-AND-INVERT topology say let us first consider **the this the** the realization corresponding to this AND-OR-INVERT. So, here you have got A B, A B this is A this is B and here A dash B dash and then this is your ah n MOS network corresponding p MOS network will be these two will be in parallel.

So, A B similarly, A dash B dash A dash B dash and you will take output from here, so this is the realizing a exclusive OR B in the in AND-OR-INVERT using AND-OR-INVERT topology, you can also realize based on this topology. In that case we will apply A in parallel with B dash similarly, A dash B connected to ground and on the corresponding p MOS network will be A and B bar will be in series **a b bar in they will be in series.**

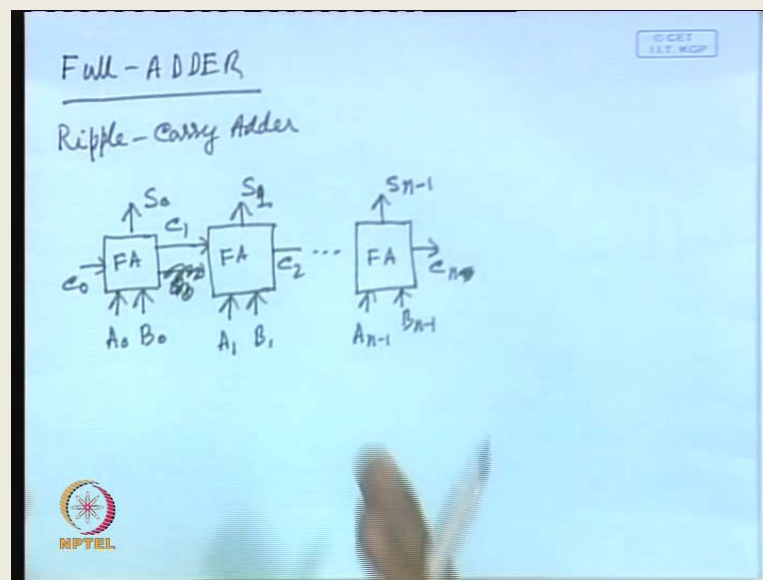
Similarly, **A bar B** A bar B will be in series and V d d is connected here ground is connected here. So, this is AND-OR-INVERT realization and this is or or, OAI OR-AND-INVERT realization. So, you can see both are possible EX-OR function has been realized using, both the topologies. So, in terms of in this particular case in terms of area in terms of delay it will be **more or less** more or less same because there are two transistors in parallel or two transistors in series. But, there is a possibility that one of the two **topo** topologies may be superior in switching characteristic, in this particular case that is not true. But, it may be it may be possible to will have realization in which one of the two topologies will be superior.

(Refer Slide Time: 38:11)



Now, we shall consider realization of a little more complex function, that is your full adder. As you know adder is one of the most common functional element, you realize you require in any circuit that performs computation.

(Refer Slide Time: 38:15)

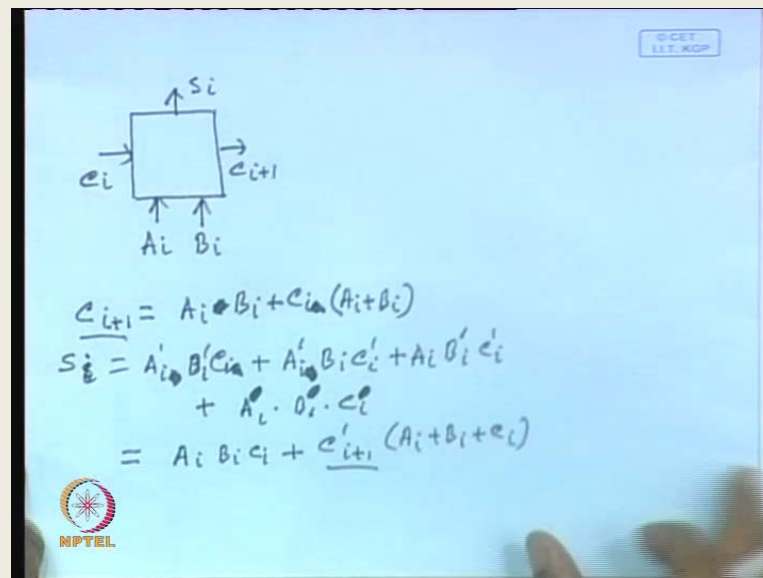


So, adder, subtractor as you know using two's complement form subtraction can be implemented by using addition. Similarly, for to realize multiplication, division you will require addition. So, adder is the most common building block of arithmetic and logic unit of any processor, so lot of research has been carried out to realize adder.

Of course, I shall consider the simplest adder that is known as ripple carry adder, what you do in ripple carry adder. Say, you apply some input A and say let us assume you are applying A 0, B 0 here, you are applying input C 0. And it produces C 1 sum and carry and that goes to the next stage **sorry** sum is available, here sum you can say is available here. S 0, **S naught** it will not go the next stage and C 1 will go there and you will apply A 1, B 1 and with C 1 as input to it and it will produce S 0, S 1 and it will produce C 2 and it will continue. Ultimately it will realize a an input adder so it will be n S n minus 1 and this is the carry generated by that stage C n minus 1 and here of course, you are applying a **sorry** it will be C n **c n** and it is a n minus 1 b n minus 1.

So you can see you **you** are using similar building block these are known as full adder, this is a full adder. So, full adder means it has got three inputs a two variables and eh carry from the previous stage and it produces carry and sum. So, if we can realize one full adder by using complex CMOS circuit you can cascade them to realize a adder of n bit adder. So, I shall see discuss about realization of a single bit full adder; single bit full adder means you will apply A i, B i and C i, minus 1 as in A i, B i, C i as input and it will produce sum and carry that means your requirement is to realize a full adder cell.

(Refer Slide Time: 41:49)



You can say where you will apply say A i B i and C i these are the three inputs and it produces two outputs S i and C i plus 1; C i plus will go as input to the next stage. So, how we can realize this, we know that the carry function C i plus 1 will be equal to A i

plus A_i and B_i plus C_i into $(A_i + B_i)$ that means whenever any two inputs are one it produces a carry.

Similarly, it produces a sum S_i , whenever any one of the inputs is 1 that means that means it is like say $A_i + B_i + C_i$ in that means if C_i is 1, one of the inputs is 1 or it can be $A_i + B_i + C_i$. That means in this case say this is complement this is complement or it can be say $A_i + B_i + C_i$ say in this case $A_i + B_i + C_i$ this is 1, this is 1. So, this is 1 or say odd number of inputs are 1, that means it can be also sum will be one whenever $A_i \cdot B_i \cdot C_i$.

And this can be simplified to represent in this form that means this $A_i + B_i + C_i$ not this I have put it wrongly, so these complements will not be here odd number of ones means three ones. So, $A_i + B_i + C_i$ or $C_i + 1 - (A_i + B_i + C_i)$. So, you can see it has been simplified and we have used the carry as the input to the next stage so this can be used to realize but, what we shall do we shall realize this in sum or product form. So, we shall realize the complement then use an inverter, so these are positive unit function. So, what we shall do we shall follow this approach that means where we have used the this kind of approach where we are using inverter at the output. So, this kind of approach we shall follow to realize this.

So, let me show it here itself (Refer Slide Time: 45:30) here for the sake of simplicity these A_i 's are not there here input is A and B input is C in and S is sum and C_0 is the carry output. So, carry in carry output and A and B are the inputs and sum is S is the sum and C_0 is the carry output. So, first the carry function has been realized in and or invert form so this is the corresponding function A and B that is in series, A and B is in series and C in and A and B in series in parallel.

And this is the n MOS network and corresponding p MOS network is the dual of this, so you have got A and B in parallel and A bar A and B these A and B are in series are in parallel with C in. And here of course, we we shall get C_0 dash not C_0 it is inferred that complement of this is realized by this network. So, this is how we have obtained the realization of the carry function and of course, you will require an inverter.

An inverter is required to produce the carry that will go to the next stage, then this is used as input to the next stage to the circuit that realizes the sum. So, here you can see

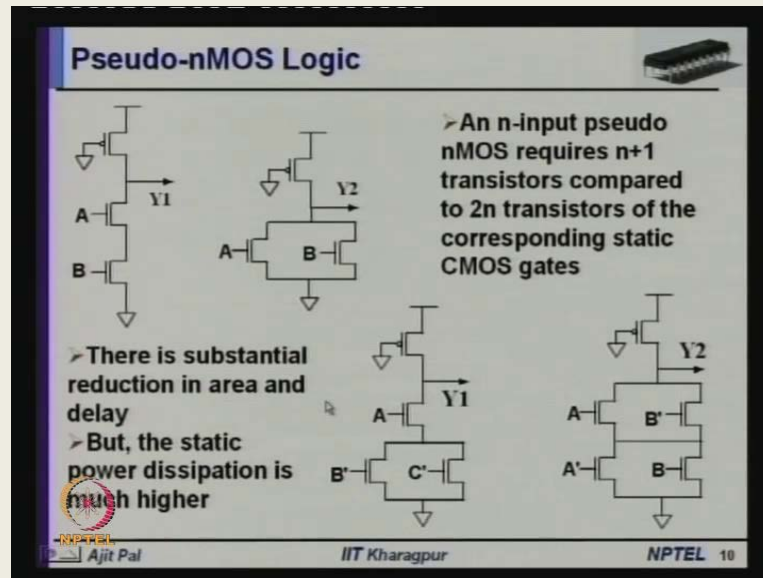
this $A A$ $A B C$ in this particular part this product term this is the corresponding network, here is the network and this C in dash network is here.

So, corresponding n mos network is shown here $A B C$ in parallel $A B C$ in parallel that is in series with $C C 0$ bar and $C A B C$ in that is in series that is in parallel to this this is there. So, parallel to this and corresponding dual is the p MOS network of course, here it is realizes the complement of sum, so you will require an inverter to realize sum S . So, how many transistors are required to realize a full adder circuit so here we see 5 plus 5 10 and this requires 3 plus 3, 7 plus 7 14 plus 10, 24 and then four requires two inverter. That means you require 28 transistors to realizes this full adder cell, so this is how you can realize a the a full adder circuit.

Now, we find that in case of full adder the total number of transistors in any gate is equal to $2n$ if n is the number of inputs you will require $2n$, $2n$ is the number of transistors considering both p MOS and n MOS transistors. And also you will also it will be next stage also the output will go to the next stage, where you have to connect it to n MOS as well as p MOS transistors. And as a consequence the CMOS circuits are very robust we have seen the, it has got very good noise margin, power dissipation is small but, it requires very large area and delay is larger.

So, what is the way out, there is there is another way of realizing CMOS circuit I have already discussed about that. That is your pseudo n MOS let us have a look at this pseudo pseudo n MOS circuit.

(Refer Slide Time: 49:14)



So here is the realization of a pseudo n MOS NAND gate and here is the realization of a pseudo n MOS NOR gate and these are the realizations corresponding **the** this is the realization of pseudo n MOS EX-OR gate.

And this is the realization of that function that I discussed, so we find that number of transistors is not $2n + n + 1$ that means, the number of transistors required in case of pseudo n MOS is not $2n$ for realizing a function of with fanning of n but, $n + 1$. So, that is a significant saving whenever you consider realization of functions with large number of fanning say hundred inputs. So, instead of 100 transistors you will require 101 transistors not only that it will drive to I mean each output will drive to only 1 transistor n MOS transistor it will not go to the p MOS transistors. Because there is p MOS transistor is there is only one p MOS transistor which is grounded.


So, we find that an n input pseudo n MOS requires $n + 1$ transistors compared to $2n$ transistors of the corresponding static CMOS gates. And this will lead to significant substantial reduction in area in delay what I am trying to tell that pseudo n MOS circuits will be faster in operation. Because **the** in the intance capacitance will be smaller and also the number of transistors will be smaller. So, area capacitance both are smaller, so pseudo n MOS circuits are faster in operation, smaller in area compared to static CMOS circuits but, unfortunately it has got a very serious limitation.

What is that serious limitation, as you know (Refer Slide Time: 51:20) when the output is zero that time this pull up net transistor is on pull down transistor is also on. And as a consequence there will be static current flow through this path, so because, of the static current flow there will be large power dissipation. So, we are discuss, we are considering we are discussing about low power circuits; obviously the pseudo n MOS circuits cannot be used for low power applications, because of large static power dissipation. So, what we can do, can we achieve best of both the worlds that means the robustness, that means the lower area of pseudo n MOS faster operation of pseudo n mos. But, lower power dissipation of c MOS can we achieve that; infact that can be achieved with the help of dynamic c MOS circuits.

(Refer Slide Time: 52:29)

Summary

- Complex logic gates can be realized using either AOI or OAI topology
- Static CMOS circuits occupy larger chip area than pseudo nMOS circuits
- Static CMOS circuits are slower than pseudo nMOS circuits
- Pseudo nMOS circuits have static power dissipation
- Dynamic CMOS circuits combine low power of static CMOS, smaller chip area and faster operation of pseudo nMOS circuits

 NPTEL
Ajit Pal IIT Kharagpur NPTEL 11

So, we can summarize what we have discussed today, we have seen that complex logic gates can be realized using either AND-OR-INVERT or OR-AND-INVERT topology we have discussed how it can be done. And we have mentioned that static CMOS circuits occupy larger chip area, than the pseudo n MOS circuits, we have also seen static CMOS circuits are slower than pseudo n MOS circuits. Because, of smaller capacitance of pseudo n MOS circuits and smaller area.

However, as I have told pseudo n MOS circuits have static power dissipation and as I have mentioned dynamic CMOS circuits, combine low power of static CMOS, smaller chip area and faster operation of pseudo n MOS circuits. So, that will be our this, I mean

that is what we shall discuss in our in next lecture. We shall discuss about dynamic CMOS circuits and we shall see how best of both the worlds of static CMOS and pseudo n MOS can **can** be combined in realizing dynamic CMOS circuits thank you.