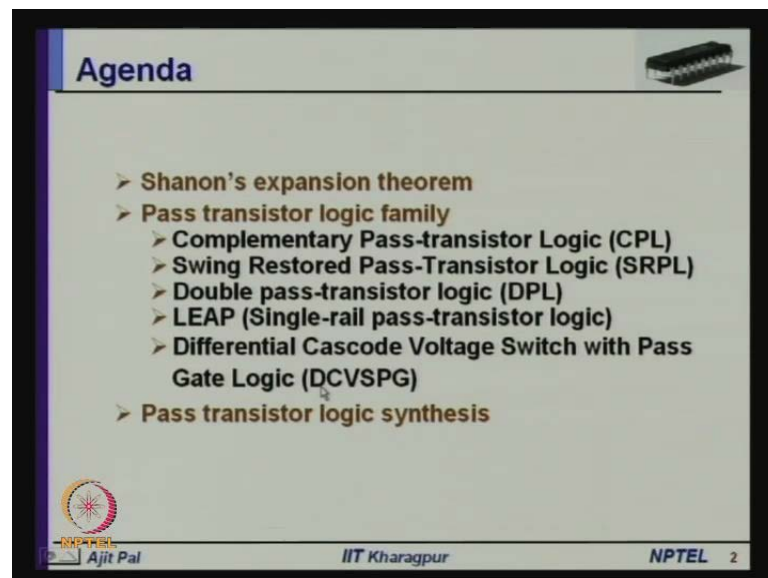


Low Power VLSI Circuits and Systems
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. # 15
Pass Transistor Logic Circuits-II

Hello and welcome to today's lecture on pass-transistor logic circuits; this is the second lecture on this topic; the last lecture, we have introduced you, the basic concepts of pass-transistor logic and discuss the advantages of pass-transistor logic circuits, and also the limitations. And we also discussed how the limitations can be overcome with the help of some additional hardware. Today, we shall start our discussion from the point, where we left in the last lecture.

(Refer Slide Time: 00:52)



So, here is the agenda of today's lecture. I shall discuss about a very important concept known as Shannon expansion theorem, and we shall see how this expansion theorem can be used in realizing the pass-transistor network. And then, I shall introduce to you a number of members of the pass-transistor logic family; first one is complementary pass-transistor logic CPL, Swing restored pass-transistor logic SRPL, Double pass-transistor logic DPL. Then LEAP which is single-rail pass-transistor logic; then differential

cascode voltage switch with pass-transistor logic DCVSPG. And finally, I shall discuss about the logic synthesis by using pass transistors.

(Refer Slide Time: 01:48)

Shanon's Expansion Theorem

- A Boolean function can be expanded around a variable x_i . $f = x_i \cdot f_{x_i=1} + x_i' \cdot f_{x_i=0}$
- Number of transistors required to realize a function is reduced

$a \cdot 1 + a'(b \cdot 0 + b' \cdot c)$ $f = a'b + ab'$

Ajit Pal IIT Kharagpur NPTEL 3

So, let me start with Shannon expansion theorem. What is Shannon expansion theorem? A function say, function f is a function of say x_1, x_2, x_n .

(Refer Slide Time: 01:57)

Shanon Expansion Theorem

$$f(x_1, x_2, \dots, x_n) = x_i \cdot f_{x_i=1} + x_i' \cdot f_{x_i=0}$$

$$f = a + be = a \cdot 1 + a' \cdot (b \cdot c)$$

$$= a \cdot 1 + a' \cdot (b \cdot 1 + b' \cdot 0)$$

© CET I.I.T. KGP

NPTEL

So, it is an n variable function which is a function of n variables; this can be expanded around a variable x_i . And it can be represented as $x_i \cdot f_{x_i=1} + x_i' \cdot f_{x_i=0}$. So, this is known as Shannon expansion, Shannon expansion theorem. So, this $x_i \cdot f_{x_i=1}$ and $x_i' \cdot f_{x_i=0}$

$\bar{a} \cdot f(x, y, z)$. What is $f(x, y, z)$? $f(x, y, z)$ and $f(x, y, z)$ these are essentially reduced functions; that means, if you expand around x , you will get $f(x, y, z)$ which is independent of x and that part of the function for which it is true for x . Similarly, $f(x, y, z)$ is also independent of x ; however, this part of the function is true when \bar{x} is one. So, this is how it can be extended; I believe, I can illustrate with the help of an example say suppose f is equal to $a + b + c$.

Now, we can expand it around anyone of the three variables; let me explain around a . So, this can be represented as $a + \bar{a} \cdot f(x, y, z)$, because you can see a and the $f(x, y, z)$ is one here, plus $\bar{a} \cdot f(x, y, z)$. Now this is the reduced function; that means this part of the function when \bar{a} is equal to one, only this part is true. Similarly, when a is equal to one, only this part is true. And of course, it is $1 + b + c$, but you know $1 + b + c$ becomes one. Now this part can again be expanded around b .

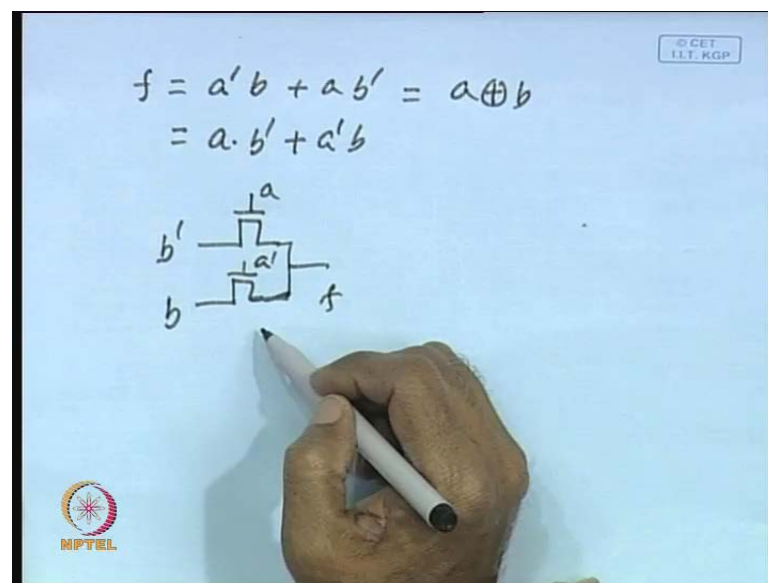
So, one of the two variables so, this can be represented as $a + \bar{a} \cdot (b + \bar{b} \cdot f(x, y, z))$; let me expand around another variable say b . So, we can represent it as, I mean $b + \bar{b} \cdot f(x, y, z)$, because the reduced function $f(x, y, z)$ is $0 + b + c$, $a + \bar{a} \cdot (b + \bar{b} \cdot f(x, y, z))$ it will be equal to sorry, it will be one and this will be $0 + b + c$; $a + \bar{a} \cdot (b + \bar{b} \cdot f(x, y, z))$ it will be $b + \bar{b} \cdot f(x, y, z)$ and $b + \bar{b} \cdot f(x, y, z)$. Because, you know it is one for when a is equal to one; this turns out to be c and when \bar{a} for \bar{b} , this part is zero. So, this is how it can be expanded.

. Now after this expansion is done, we can actually map it to pass-transistor logic. So, this part is logic independent of how it is being realized. Now this part after the expansion is done, this can be used to realize the circuit using pass-transistor logic. For example, we start with expansion with a . So, here we apply a to this part so, and this is one. And then, if we expand around \bar{a} , we have got two components. And this is being expanded around b so, it will be equal to b and then this is c and when is expanded around \bar{b} , it is zero. So, this is the realization of, this is the realization of the pass-transistor logic.

Earlier we obtained the realization of this by using multiplexer; there we have seen it can be used to realize this function, can be realized using multiplexer. However, whenever we use multiplexer to realize a function, you require more number of transistors. For example, this particular realization of this f will require how many, how many transistors? There will be four inputs, if we expand around a and b .

And this is the output f. So, it inside you will require 1 2 3 4 5 6 7 8; 8 transistors will be required inside it. And here of course, you have to apply the other the various components after expanding around a b, a and b. That means from the truth table, we can find out that will hear it will be, you will be applying here g 0, g 1, g 2 and g 3. So, instead of requiring eight transistors inside this pass-transistor logic network, you require only 1 2 3 4. So, instead of eight transistors, you require four transistors. So, this is how you can carry out expansion of a function and then map it to the pass-transistor network; pass-transistor network means nMOS transistors. So, this is how it can be done.

(Refer Slide Time: 08:10)



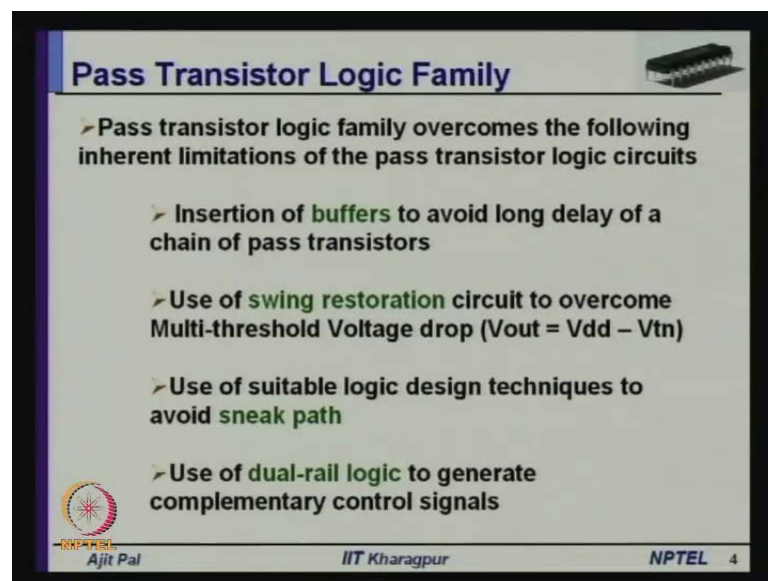
Now let me illustrate with the help of another example. Suppose f is equal to a bar b plus a b bar; typical, I mean this is the exclusive or function exclusive or b. Now this can be expanded around a; so, it will be a b bar. It is already in expanded form and a dash b. So, what will be the pass-transistor network based on this? It will be here you will require a and a bar, a bar and b. So, this is how only by using two transistors, this realizes f; this f this particular f which is a exclusive or function, just you require two transistors. On the other hand if you perform you know, if you realize by using multiplexer function user you will require more number of transistors.

So, this is the, the advantage of this Shannon expansion theorem; that means using Shannon expansion theorem, you can get a pass-transistor network which will require minimum number of transistors. Of course, later on we shall see this ordering of variable

here for example; we have first expanded around a then expanded around b. So, if you have got n variables, you can expand around any one of the variable then another one from the remaining variables and so on. This ordering has an important role in the expansion; particularly, the expanded form will be heavily dependent on the ordering.


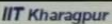
And then it has been found that getting an optimal solution after expansion is a np-complete problem; that means which particular ordering, you will give you a minimum number of transistors; that is an np complete problem. However, there are some heuristic based techniques by which the expansion can be done and a good ordering can be made. Such that you can get transistor with, you can realize minimum number of transistors. So, this is Shannon, Shannon expansion theorem. Later on, we shall be using this in our, in our future circuits.

(Refer Slide Time: 10:29)



Pass Transistor Logic Family

- Pass transistor logic family overcomes the following inherent limitations of the pass transistor logic circuits
 - Insertion of **buffers** to avoid long delay of a chain of pass transistors
 - Use of **swing restoration** circuit to overcome Multi-threshold Voltage drop ($V_{out} = V_{dd} - V_{tn}$)
 - Use of suitable logic design techniques to avoid **sneak path**
 - Use of **dual-rail logic** to generate complementary control signals

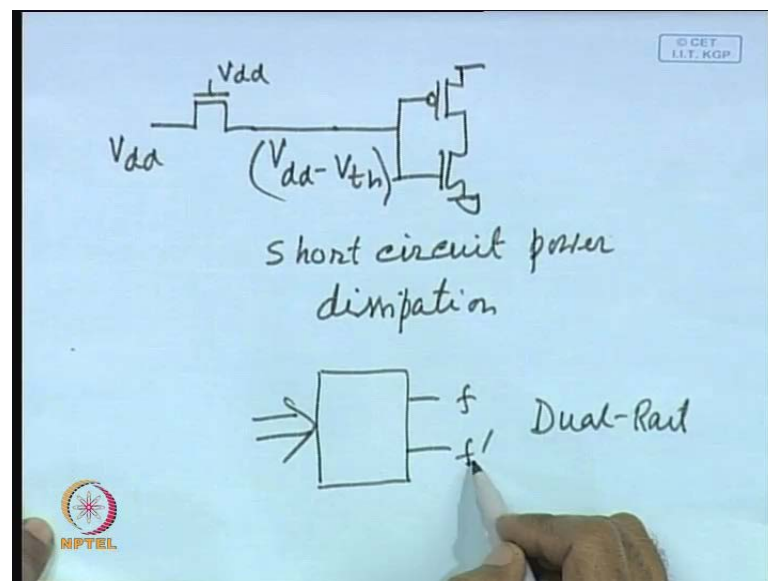
 Ajit Pal  NPTEL 4

Now I shall introduce to you the pass-transistor logic family. Obviously, the pass-transistor logic family has been, I mean has been provided, has been has been merged such that, it overcomes the various limitations of pass-transistor logic circuits. We know that there are a number of inherent limitations of pass-transistor logic; these inherent limitations are overcome with the help of these logic families and I shall introduce to you various members of this family.

What are the limitations that you have to overcome; first limitation is insertion of buffers to avoid long delay of a chain of pass transistors. As you know, as the number of

transistors increases in series, the delay increases; more or less quadratically at the rate of a square. And as a consequence after when the number of pass-transistor stage or number pass transistors in series is large say, eight or ten or fifteen then delay can be unacceptable. So in such a situation, you can put buffers as we have discussed in the last lecture so, all the family members. We will include a buffer at the output and such that, the long delay can be avoided. Second limitation was you know that there is some multi-threshold voltage drop at the output.

(Refer Slide Time: 12:15)



So, you have to use some swing restoration circuit to overcome multi-threshold voltage drop. As we know, whenever the signal passes through a pass single or multiple pass-transistors. Even if you apply V_{dd} here; you will and the gate voltage is V_{dd} , you will get V_{dd} minus V_{th} .

And if this is applied to one gate logic say an inverter, then what will happen? It will not only get lesser drive, but this voltage may be such that both the transistors are on. So, it will lead to what is known as short circuit; Short circuit power dissipation.

So, to avoid this short circuit power dissipation and lesser drive, it is necessary to restore the output to the V_{dd} level and that is done with the help of swing restoration logic. And I have introduced to you the type of circuit you require to restore the voltage level with the help of swing restoration logic. Third limitation that we encountered is the

problem of sneak path. As we know, whenever the output gets a path to one as well as zero simultaneously, this leads to what is known as sneak path.

Actually sneak path can be avoided by proper design of the pass-transistor network. And we have already discussed about the pass realization of net pass-transistor network using Shannon expansion and whenever you realize the pass-transistor network using Shannon expansion theorem, you will find this kind of sneak path cannot exist. So, sneak path is can be avoided by using proper design, using particularly using Shannon expansion theorem. Later on we shall discuss about another technique use of binary decision diagram B d d which can also be use to realize pass-transistor network and that will also avoid sneak path.

Fourth problem as we know that the it requires both x and x bar, I mean both the complementary and uncomplimentary inputs are required as input; that is the reason why the pass-transistor logic circuits are inherent or inherently dual value; dual-rail in the sense that means, you will realize a circuit in such a way. It will produce not only f , but also f bar. We will apply input, primary input and it will produce f and f bar. So, this is known as dual-rail so, dual-rail is necessary. So, that you can feed both x f and f bar which is necessary as we know. For example, even whenever you are realizing this type of circuit, you require a and a bar, b and b bar.

Similarly, the later power of the circuit will require f and f bar; that is the reason why you have to realize both and f and f bar. Such that, the pass-transistor network can be driven by a , the by these two signals, so, these are the features which will be included as part of your pass-transistor logic circuits.

(Refer Slide Time: 15:38)

Pass Transistor Logic Family

> Complementary Pass-transistor Logic (CPL)

- > Inverting buffers perform restoration of logic levels at the output
- > Inverting buffers allow driving large capacitive loads
- > The pMOS latch performs swing restoration
- The pMOS devices should be properly sized so that the circuit can function correctly

NPTEL Ajit Pal IIT Kharagpur NPTEL 5

So, let me introduce the first family member; that is your complementary pass-transistor logic CPL. So, CPL or complementary pass-transistor logic, CPL as the as its name suggests complementary, complementary means it has got both the outputs, complementary outputs; that means, f and f bar; that means, it is dual-rail.

(Refer Slide Time: 15:55)

CPL

$$A \cdot B$$

$$\overline{A \cdot B}$$

$$A \cdot B = A \cdot B + A' \cdot 0$$

$$= B \cdot A + B' \cdot 0$$

$$\uparrow$$

$$B$$

$$A' \cdot B = B \cdot A + B' \cdot B$$

$$\overline{A \cdot B} = B \cdot A' + B' \cdot B'$$

© CET I.I.T. KGP

NPTEL

So, what is being done? Two networks are used; one is your one for realizing f; another for realizing f bar. So, it will have a one pass-transistor network and you will require two pass-transistor networks to realize the complementary function. So, another pass-

transistor networks, another pass-transistor network to realize f and \bar{f} . So, you will apply inputs to both, to both of them and from there you will get f and \bar{f} .

However in addition to this, it will insert buffer in the form of inverters. So, inverting buffers are provided at the output. In addition to that two weak p most transistors are used; this is connected to V_{DD} . Here also, another weak p most transistor is there; this is connected to V_{DD} and instead of grounding them, this is connected to the, to this point. As we know, this is the complementary output of this one. So, that is the reason why instead of connecting from inverter output, it is it is connected from here. Similarly, this is this is connected to this.

So, this is the generalized structure of a complementary pass-transistor logic circuit which is one of the, one of the members of the pass-transistor logic. So, here you realize f and here you realize \bar{f} .

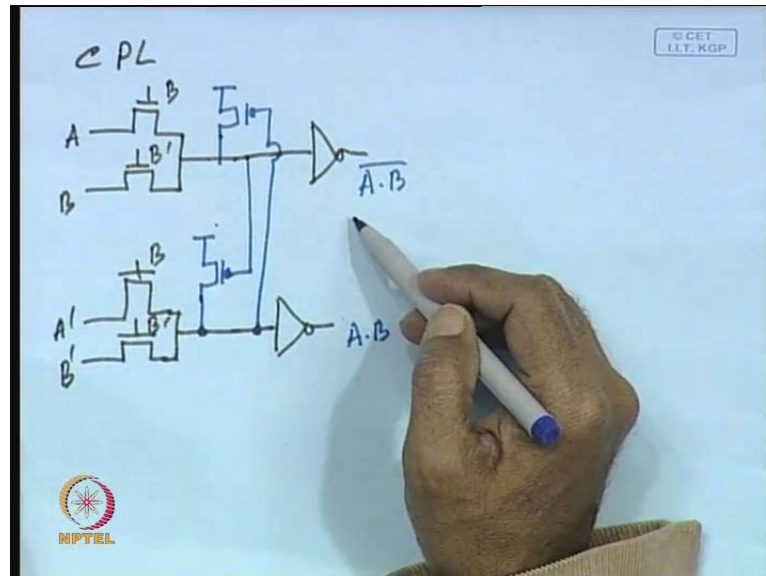
Let us consider the realization of some real functions say suppose you have to realize AND and NAND. How do you realize this pass-transistor network? Since A and B , you see this is an AND function, **AND function**; this is an NAND function. So, you can again use the Shannon expansion theorem here. So, $A \cdot B$ can be represented as $A \cdot B$ plus $A \cdot \bar{B}$; $A \cdot \bar{B}$ sorry, it will be $A \cdot 0$ or if you expand around B , it is not necessary that you have to always expand around A . If you expand around B then, it will be $B \cdot A$ plus $B \cdot \bar{A}$. Now instead of applying 0 here, what you can do sometimes us instead of applying a constant, we apply some variable so, you can put $A \cdot \bar{B}$ here. As you know, when B is one, $B \cdot \bar{A}$ will be 0. So, this will realize the same thing.

So, based on this you can realize the network of this pass-transistor logic, similarly for this complementary function. One particular property of this pass-transistor logic is that you can see say suppose you expand around B , so it is $B \cdot A$ plus, let us assume $B \cdot \bar{A}$. This is how you realize $A \cdot B$.

Now without doing any expansions $A \cdot \bar{B}$ that is NAND function can be obtained simply by putting these inputs in the complemented form. That means, that $B \cdot \bar{A}$ plus $B \cdot \bar{A}$ will realize this complementary function. Because, you will see that this realizes the complementary functions so, we do not have to do expansion only thing that you have to do these inputs; those inputs which you apply to the pass-transistor logic,

they have to be complemented. Then it will realize the complementary functions as we shall see in the realizations.

(Refer Slide Time: 20:47)



Now with this, let me let us realize the NAND and NOR functions for the CPL version of NAND and NOR. So, you will require two transistors for realizing the complementary and these functions. So, you require, you will require if you expand around B A, and A and B there will be the inputs. So, A whenever it is B and B bar B and similarly, here B and B bar. So, we have to apply the complement of this. So, it will A bar and here will be B bar. And then, this we will go to, as we as we know it will go to an inverter; this will go to another inverter.

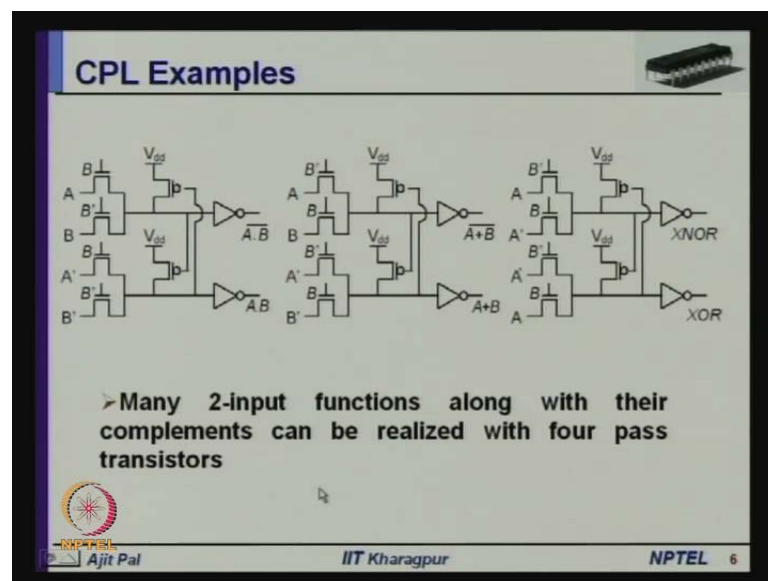
And here, you will require two pMOS transistors for swing restoration, another pMOS transistor that will be connected, gate will be connected there; this should be connected to V d d and this will be connected here. So, this realizes as we know, this is this is the output of AND and so, it will be NAND here; A dot B bar. And here, it will require A dot B. Here it is NAND so, it will be AND here. So, this is how you can realize and, and with the help of this CPL.

Now what are the advantages and disadvantages of this? We have seen that in this CPL logic inverting buffers perform restoration of logics levels at the output. So, we have inverting buffers at the output and inverting buffers allowed driving large capacitive loads and pMOS latch you see since they are connected back to back. This output is

connected as to the base to the gate of this transistor; this output is connected to the gate of this transistor; they form a kind of latch. So, these two pMOS transistors perform swing restoration.

In this particular case, one aspect you have to remember that the pMOS devices should be properly sized. So, that the circuit can function correctly; you see the sizing of the pMOS transistor is important, because they will decide how quickly the switching will take place and also, what kind of power dissipation that will occur. And of course, in this particular in the steady state, it will not affect the output, because since it is acting as a kind of latch. The output voltage levels will be always 0 and V_{dd} in both the, for both the outputs. So, here it will be V_{dd} and 0 depending on high or low and here also, V_{dd} and 0 depending on high or low.

(Refer Slide Time: 24:02)



However proper sizing of this pass transistors is necessary, to avoid unnecessary power dissipation and delay. And using the same concept we can realize different types of gates by using this CPL technique, you can see here OR and NOR has been realized and in the same way, you can realize the pass-transistor network. And here, we have realized the XOR and XNOR again by realizing the p, the pass-transistor network using Shannon expansion theorem.

(Refer Slide Time: 24:42)

Pass Transistor Logic Family

➤ Swing Restored Pass-Transistor Logic (SRPL)

The slide contains two circuit diagrams. The left diagram shows an nMOS CPL network with two inputs and two outputs, O' and O , connected to two cross-coupled CMOS inverters. The right diagram shows a 2-input NAND gate realization using two pass-transistor networks (top and bottom) and two cross-coupled CMOS inverters. The top network has inputs A and B and output $A \cdot B$. The bottom network has inputs A' and B' and output $A \cdot B$.

➤ The cross coupled CMOS inverters restore the logic levels
➤ SRPL realization of 2-input NAND gate is shown
➤ Sizing is critical for speed and power dissipation issues

NPTEL Ajit Pal IIT Kharagpur NPTEL 7

So, this is how you can realize different gates with the help of CPL technique. Now coming to the second family member, swing restored pass-transistor logic.

(Refer Slide Time: 24:55)

SRPL

The diagram shows two pass-transistor networks connected to a cross-coupled CMOS inverter pair. The top network is labeled f and the bottom network is labeled d' . The output of the inverter pair is V_{dd} . The diagram is annotated with the following notes:

- Dual-Rail
- Swing Restoration Buffer
- Less isolation

© CET I.I.T. KGP.

So, in swing restored pass-transistor logic so, this is SRPL, swing restored pass-transistor logic. In swing restored pass-transistor logic, what is done? Again two pass-transistor network is used like the previous case. So, you have pass-transistor network **pass-transistor network** and you will require two pass-transistor networks. It may be noted that the number of transistors required in each network will be same; only difference will be

the inputs will be different; that means, complementary of the other; that means number will remain same. And here these outputs were the buffers are not provided at the output, but they are connected in this form, back to back.

The inverters are connected back to back. So, here you will apply the primary inputs in the same manner, in the same way you will apply the primary inputs here and the outputs should be taken from these two points. So, it will be f and this will be \bar{f} . It is so in this case, we are finding that two inverters are connected back to back. How it really performs various functions like you know number one, it is dual-rail, dual-rail as that we have seen, because we are realizing f and \bar{f} .

Now, how it is performing the role of swing restoration? How the swing restoration is done? We can see here, this inverters I can expand one of the two inverters. As we know this is the static CMOS inverter. So, you will have a pMOS transistor and nMOS transistor and this is actually equivalent to this inverter; it is connected to ground; this is connected to V_{dd} . Now let us assume this particular, this particular inverter is this one.

Now, base is connected here and output is connected here. So, base is connected here and output is connected here. Now you see this particular pMOS transistor will act as swing restoration of this one. How? Let us assume this is zero so, this zero means, this output is supposed to be one. And so, since this is 0; this transistor will be on and this will provide, this will restore the logic level to V_{dd} . So, the swing restoration is done by the pMOS transistor of this inverter for this particular output. Similarly, the swing restoration for this output is done by the pMOS transistor of this particular inverter. So, you see that swing restoration is done with the help of the two pMOS transistors of the two inverters. We do not require any additional transistors for that.

Third requirement as we know, we have to insert buffer. Here you know, for this output this inverter is acting as a buffer. We can see it is driving so, you can drive large capacitive load. Similarly for this output, this is acting as a buffer; this inverter is acting as a buffer. So, you find that the two inverters performing dual-rail; the dual-rail is first of all, they are doing the swing restoration. They are also doing theirs, they are also performing the role of buffer; that means, buffer means whenever you have to drive large capacitive load, some buffer is required.

That will reduce the delay as we know, buffer insertion is necessary to reduce delay. So, they will help to reduce the delay as well as they will perform the swing restoration; however, you are duplicating the pass-transistor network to get dual-rail output.

So, here is the, here one NAND and NOR version of the SRPL circuit is shown here; it is not different from the other logic family; this nMOS network, that pass network is identical expect, I mean only difference is the output path, driver path. So we see, we have got inverters connected back to back and we are getting $A \cdot \bar{B}$ and $A \cdot B$.

Now you may be asking is there any basic difference between this and that CPL? In what way they differ? In case of CPL, the inverter was provided at the output, but here inverter is not provided at the output; that means, the pass-transistor logic was not driving the output in case of CPL as you can see.

So, CPL pass-transistor logic is driving the inverter and inverter output is driving the output. But in this particular case, you can see the directly the pass-transistor logic is driving. Of course, that inverter is there which is acting as swing restoration logic and it will definitely give some drive, because opposite polarity input is available here.

Now, what is the outcome of this? Outcome is that you can see, this B is as if B is transmitted through this network to the output so, isolation is less here. In case of CPL, inputs are applied to the gate and you know to the, you know to the input of the pass-transistor logic. It is usually source or drain, because they are interchangeable and that like gate logic, we are taking the output from the output of an inverter, but here you are directly taking from output of the pass-transistor network. So, here we can say that less isolation; that means, whenever isolation is less, the input signal will noise and disturbances of the input will pass to the output; it will not be, it cannot be suppressed by the inverter driver. So, isolation is less in this particular case.

Now and here also, sizing is critical for speed and power dissipation issues; the sizing of the transistors of these inverters is critical. You have to huge proper width of these transistors such that, the power dissipation is not high and speed of operation is not much affected. It will definitely effect to some extent, but you have to size it them properly. So, that necessary speed and power dissipation is achieved.

(Refer Slide Time: 32:50)

Pass Transistor Logic Family

➤ Double pass-transistor logic (DPL)

- The DPL is a modified version of CPL
- Both nMOS and pMOS logic networks are used together
- As this provides full swing on the output, no extra transistors for swing restoration is necessary
- It has balanced input capacitance

Ajit Pal IIT Kharagpur NPTEL 8

Coming to the third family member, which is known as double pass-transistor logic DPL; in case of DPL, again you are using dual pass-transistor logic DPL; you are using both nMOS and pMOS transistors earlier we have seen.

(Refer Slide Time: 33:03)

DPL

$A \cdot B = A' + B'$

© CET I.I.T. KGP

NPTEL

So, far as if pass-transistor is concerned, you were using only nMOS transistors. But in case of DPL, you will see both nMOS and pMOS transistors are used and typically the

nMOS transistors are used to provide a path to the ground. On the other hand as you know, pMOS transistors are used to provide a path to V_{DD} .

The reason for that are nMOS transistors you know is not a, does not pass high level signal properly. Similarly, pMOS transistor does not pass low level signal properly. So, here the same concept has been used, but the configuration is different from the static CMOS. We are using pMOS in the pull-up network and nMOS transistors in the pull-down network, but not the way it is done in case of static CMOS; it is different. Here they are connected like pass transistors as you can see. So, here $A \cdot B$ so, $A \cdot B$ you know, $A \cdot B$ can be realized by this network.

So, let me explain say $A \cdot B$; here it can be represented by $A \cdot B$ or $B \cdot A$. So, this is, this is how you can realize the pull-down network; that means, you can see you can realize $A \cdot B$ and this is connected to ground.

That means, when any of them is one; that means, $A \cdot B$ is equal to, $A \cdot B$ is A is equal to 0 or B is equal to 0 as you know in case of NAND gate, if any of them is gate is zero, output will be connected to, output will be zero.

And that means, if any of the input is 0 this will be one so, this will be connected to zero. So, this is, this will realize the $A \cdot B$ NAND function and what about the pMOS network? pMOS network is realized in this way.

Here B bar, in a similar way you are using B bar and A bar these two are connected; however, here you will be putting A and B and this will realize the $A \cdot B$ function. In a similar way you can realize the complementary functions; this is a NAND and NOR function. And sorry, this is the NAND function you can realize in this manner and in this case, this is connected to V_{DD} and this is connected to... this is your $A \cdot B$ and this is $A \cdot B$ bar and this is connected to $A \cdot B$.

A is connected to, these are connected to V_{DD} and here you were realizing $A \cdot B$. So, when both of them are one, then output will be one; both of them are one means B is equal to one, B is equal to one and A is equal to one I mean, A is equal to one, B is equal to one; A is equal to one, B is equal to one; both of them are one; then output will be one. No this is I think, I think this is NAND, this will be AND; this is AND this is NAND (0). So, this is how you can realize and and functions by using DPL. So, the DPL is

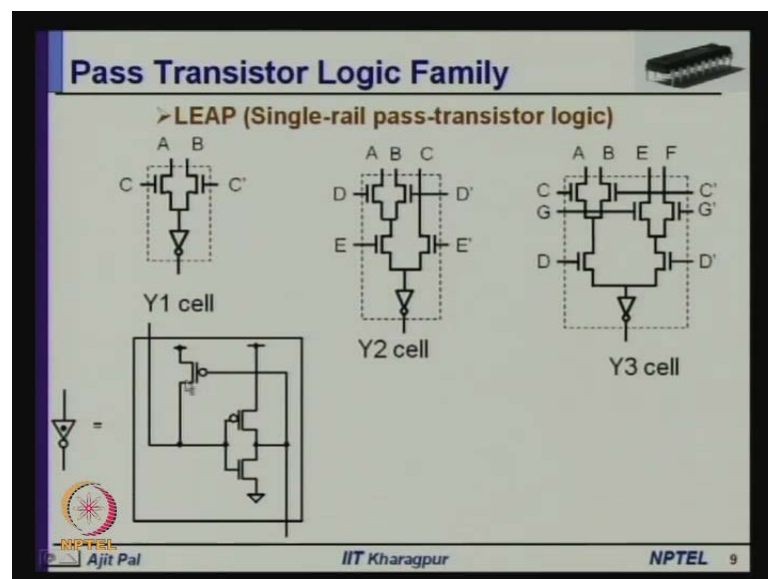
modified version of CPL; in case of CPL we have seen two pass transistor logics are used, but here you are replacing this part and this you are replacing the pass-transistor logic part by using pMOS transistors.

And both nMOS and pMOS logic networks are used together. As this provides full swing on the output; no extra transistors are required for swing restoration. So you can see, since nMOS transistors are used for, for making it zero and similarly, these transistors are used to make the output one; you will always get good logic level outputs. So, there is no needing swing restoration logic at the output. Here also, you do not require any swing restoration logic.

And another very important feature is that, it has balanced input capacitance, you see number of transistors that is connected here. For example, A here one, here also A is one; A bar is one, here A bar is one here; that means, both the inverting and non-inverting part will require same number of transistors.

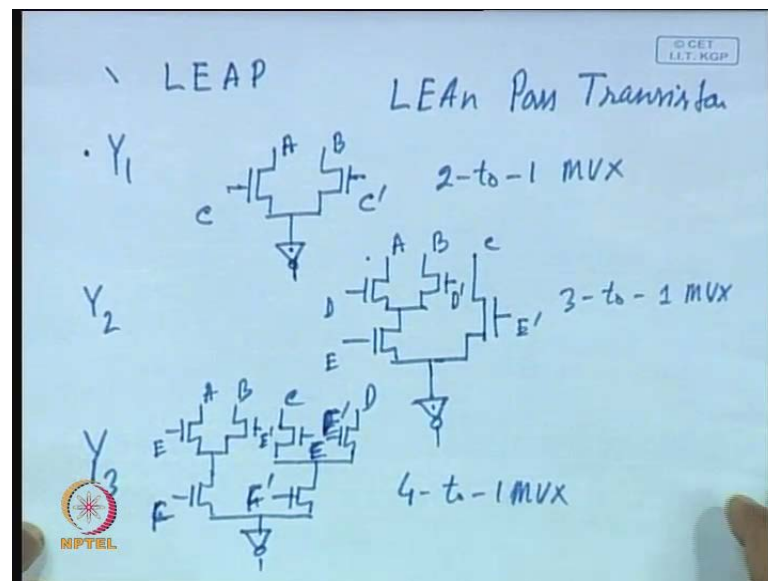
As a result, the capacitance is balanced for the, for the two different networks and this will give you switching characteristic, Ideal switching characteristics of both the outputs. So, switching characteristics will be identical, because of the balanced input capacitance of this double pass-transistor logic; this is the third member DPL.

(Refer Slide Time: 39:25)



Now let us consider the fourth member LEAP. So, LEAP actually stands for lean pass-transistor logic; the reason for that it is called lean. It is thinner, because you will be using single-rail logic in contrast to dual-rail logic that is used for other pass-transistor members, logic family members. Here you will see that, the any circuit is realized with the help of three cells; these are known as y one cell, y two cell, and y three cell.

(Refer Slide Time: 40:04)



So, there are three cells; LEAP - LEAn Pass-transistor logic. So, leap has come from this LEAn, LEAP lean pass-transistor logic, because number of transistors required is smaller because, it is not dual-rail. Now three cells are there; one is known as y one; second one is known as y two; third is known as y three. Actually if you look closely, you will find that, this y one is nothing but a 2 to 1 multiplexer. So, here you apply C, C bar and two inputs are available A B. And here of course, you will provide some buffer at the output. We shall see what kind of buffer is that. So, this is nothing, but a two to one multiplexer.

We have got two inputs and C is the control input and you have got a buffer circuit at the out inverting buffer at the output. So, this will also do swing restoration as we shall see. So, y one is nothing, but a two to one mask multiplexer

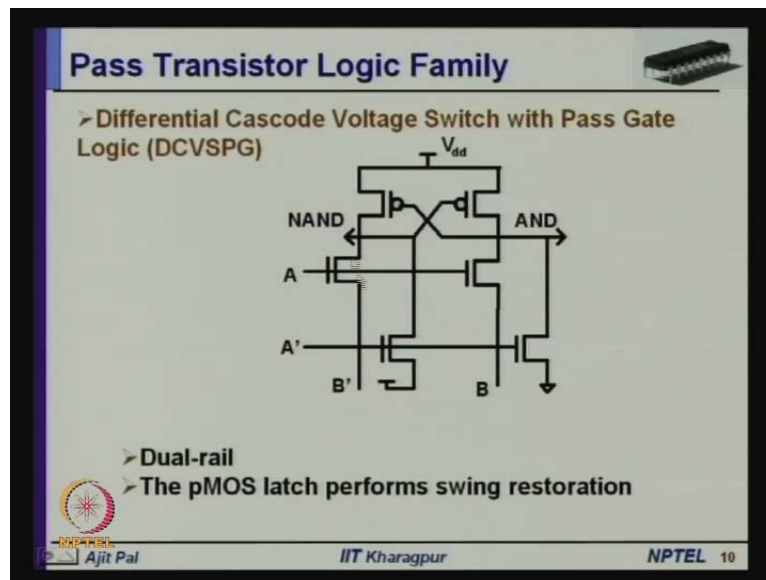
Similarly, this y two is essentially three to one multiplexers. So, it has got two to one and another transistor is there, again this is connected to that buffer inverter. So, here you have got three inputs A, B and C and D, D bar, E, E bar. So, D and E are control signals

and these three are inputs and there is buffer at the outputs. So, this is nothing but, three to one mask.

So, in a similar manner y three can be realized which is essentially a fourth to one mask. So, you have got A, B, C, D sorry, C and D and here we will apply A, B, C, D, E, E bar, F, F bar. These are tied together; then this is connected to E, F, and G here. There will be E, F, G and A, B, C, D, E here actually E and E bar here also use E and E bar and F and F bar connected here; F bar and buffer is there. So, this is nothing, but a four to one mask

And then the swing as it is shown here, the same circuit y one, y two, and y three cells - two to one, three to one, four to one multiplexers and here is the buffer. Buffer as it is shown is nothing but an inverter and a weak pMOS transistor for swing restoration. So, this path performs the dual-rail of providing buffer as well as swing restoration logic. So, this kind of buffer is provided at the output of each of the cells. Now any circuit can be realized in terms of these cells y one, y two, and y three cells and in the number of the cells required is only three.

(Refer Slide Time: 44:11)




So, this is the fourth member of the logic family coming to the last member of the logic family which is differential cascode voltage switch with pass gate logic DCVSPG.

Here two pMOS transistors are connected back to back in the form of a latch; then you have got two pass-transistor logic same kind of pass-transistor logic. So, here this one is


AND at the output you will get AND at the here at the output you will get NAND. So, the pass-transistor realization can be done in the same way and only difference is that instead of inverter, you have got two pMOS transistors which are connected back to back and name is differential cascode voltage switch with pass gate logic. So, and as you can see, it is, it has got dual-rail, two outputs, the pMOS latch performs swing restoration.

So, these two transistors are also doing swing restoration and however, isolation is less here, because pass-transistor logic is directly driving the outputs. So, there will be less isolation.

(Refer Slide Time: 45:17)

Comparison of the Pass Transistor Logic Families 

Logic style	#MOS networks	Output driving	I/O decoupling	Swing restoration	# rails	Robustness
CMOS	$2n$	Med/ good	Yes	No	Single	High
CPL	$2n + 6$	Good	Yes	Yes	Dual	Medium
SRPL	$2n + 4$	Poor	no	Yes	Dual	Low
DPL	$4n$	Good	Yes	No	Dual	High
LEAP	$n + 3$	Good	Yes	Yes	Single	Medium
DCVSPG	$2n + 2$	Medium	Yes	No	Dual	Medium


IIT Kharagpur
NPTEL 11

Let us quickly compare the various logic families. Here as you can see, we have five members of the logic family compared with static CMOS. The number of transistors that is required is shown here.

CPL as you know, CMOS require two n transistors, if n is the number of inputs number of variables; CPL will require two n plus six, because inverter and the weak pMOS transistor three plus three six. SRPL will require two n plus four, because you do not require the weak pMOS transistors, but only two inverters are required at the outputs so, two n plus four. DPL requires four n ; you do not require neither inverter nor swing restoration pMOS transistor, but you require two n for each network. So, you require four n transistors to require dual-rail circuit.

LEAP requires n plus three, because the n is the number of inputs plus three which is essential that driver and swing restoration logic. And DCVSPG requires two n plus 1 transistor.

So, output driving capability as you can see, it is medium to good for CMOS; it is good for CPL, because you have got explicit inverters at the output. So, you get good output driving capability. But it is poor for SRPL as we have seen the, in case of SRPL, the pass transistors are directly connected to the output.

Similarly, for DPL it is good, for lean LEAP it is also good you have got pMOS. The inverters at the output for DCVSPG, it is again medium. The reason for that is, here also pass transistors are directly connected to the output. So, for as I/O coupling is concerned, there is no I/O coupling in SRPL, I mean I/O decoupling; there is I/O decoupling; that means, inputs are inputs are not, pass transistors are not directly connected to the output.

This is directly connected to the output for SRPL, but not in other cases. I think in case of DCVSPG also it is connected to the, directly to the output. Here also, it is already connected. So, but you have got pMOS transistors connected that is why in this case also I/O decoupling will not be good so, here will be no. Then swing restoration logic, you do not require in case of static CMOS and you also do not require for DPL and DCVSPG. But you require explicit swing restoration logic for CPL, SRPL and LEAP logic family members and except static CMOS and LEAP all are dual-rail as you have seen.

And so far as robustness is concerned, static CMOS is very robust; DPL is also very robust, but others are not as robust as, static CMOS and DPL others are moderately or low robust.

So, this is how, this is a relative comparison of the pass-transistor logic families. Now we are the pass-transistor logic family is now known to us. Question is how we shall realize logic circuits using these pass-transistor logic families.

(Refer Slide Time: 48:59)

Pass Transistor Logic Synthesis

- Two phases of synthesis:
 - Technology independent optimization phase
 - Get ROBDD of a given Boolean function
 - Technology mapping phase
 - Map the BDD nodes onto PTL cells

BDD → PTL tree

NPTEL Ajit Pal IIT Kharagpur NPTEL 12

Actually, the logic synthesis State has got two states; first or you can say, there are two phases; in the first phase, you do technology independent optimization phase and one commonly used technique is get ROBDD of A given Boolean function. ROBDD stands for, ROBDD stands for Reduced Ordered Binary Decision Diagram, BDD. This and actually this technique that we discuss Shannon expansion technique; that is also a logic independent optimization technique; there we have seen, we are doing an expansion and we are getting a network.

That is also a logic independent optimization technique and second technique; second phase is mapping the BDD nodes onto the PTL cell. So, these are the two phases so, it is like this. So, you get a BDD and in the next phase, you will do the mapping on the pass-transistor logic tree.

Now, the pass-transistor logic tree can be mapped depending on which particular logic family, you are using; you can use CPL, you can use DPL, you can use LEAP, you can use SRPL. So, that part is logic dependent. On the other hand, the BDD part is logic independent.

(Refer Slide Time: 50:30)

ROBDD - Construction

- Construction of a Reduced Ordered BDD
- Canonical representation of a Boolean function (ROBDD)
- Convenient data structure for Boolean logic representation and manipulation

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Truth table

$f = ac + bc$

Decision tree

Legend: — 1 edge (f_x), - - - 0 edge ($f_{\bar{x}}$)

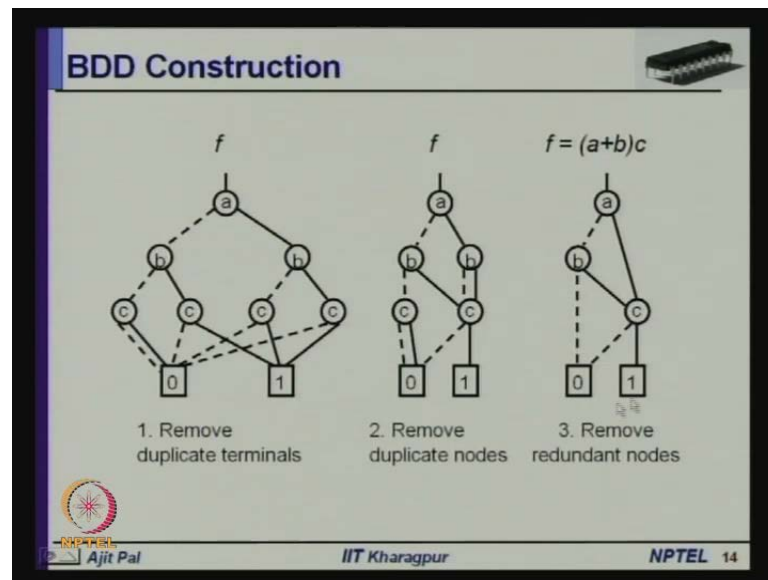
NPTEL Ajit Pal IIT Kharagpur NPTEL 13

Let us discuss, how you can get a ROBDD construction? This ROBDD is a canonical representation of a Boolean function; that means, what do you mean by canonical representation? Canonical representation is unique; for a given function, it will be unique that is why it is called canonical representation.

And it is a convenient data structure for Boolean logic representation and manipulation and that is the reason, why BDD is widely used for synthesis verification of functions. So, BDD plays a very important role in logic synthesis and verification.

Let us discuss how BDD can be obtained. So, for this particular function f is equal to $a c$ plus $b c$, if from the truth table you can you can construct a decision tree. Decision tree you know, you can choose take one variable say a . We have started with variable a ; then it has got one edge and zero edge; one edge is for which the reduced function, for which reduce, for which a is one. And here, this part will be essentially, the reduced function for which a is zero; that means, essentially we are doing kind of Shannon expansion here. So, this is the $f a$, this is $f \bar{a}$. So in this way, if you do the expansion, you will get a tree like this. And at the LEAF node, you will get zeros and ones essentially, it can, you can take it from the truth table so, 0 0 0 1 0 1 0 1. So, I have put them there.

(Refer Slide Time: 52:13)



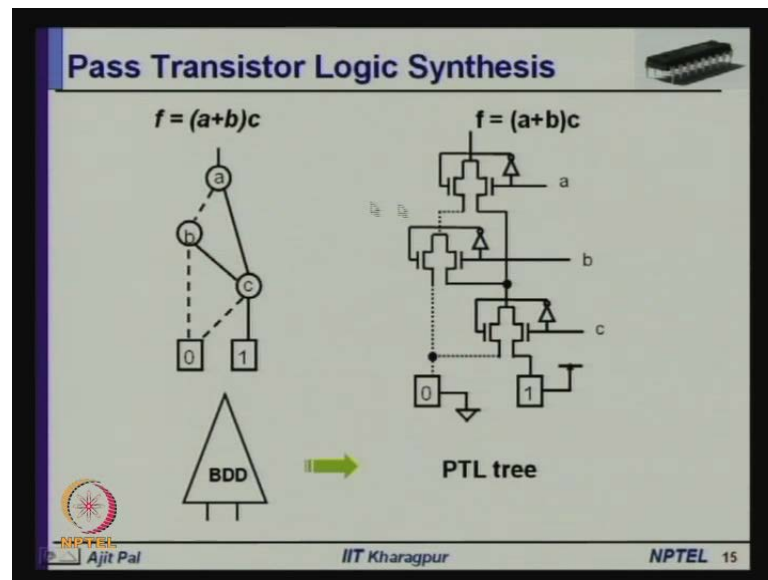
Now you can this is the decision tree, now you have to do a kind of optimization or reduction. First thing that is being done; all these zeros are combined and all these ones are combined to have only two leaf nodes; 1 for 0, another for 1.

So, that is what is being done by removing duplicate terminals. So, here duplicate terminals are reduced, removed from the output. Then here again, you can see here c; we have removed where their identical. For example, in this particular case b, whenever it is coming this is, this c, you can see b dot c; this is coming to 0; b c it is also coming to 0. So, they are combined here b c and here you have got only two c's and this is how we have removed the number of nodes for c. So, only two nodes are required for c.

And finally, we shall reduce, remove the redundant nodes. For examples, this c is not required even for 0 as well as 1; it is connected to 0 obviously, this is redundant. Similarly, this b is redundant, because both it is connected for 0 as well as for 1. So, this is also removed. So, both of them are removed and getting a redundant I mean, remove the after removing the redundant nodes, we get a final tree. And this is actually the reduced ordered BDD. Reduced, why it is reduced ordered BDD? We have removed unnecessary nodes and edges.

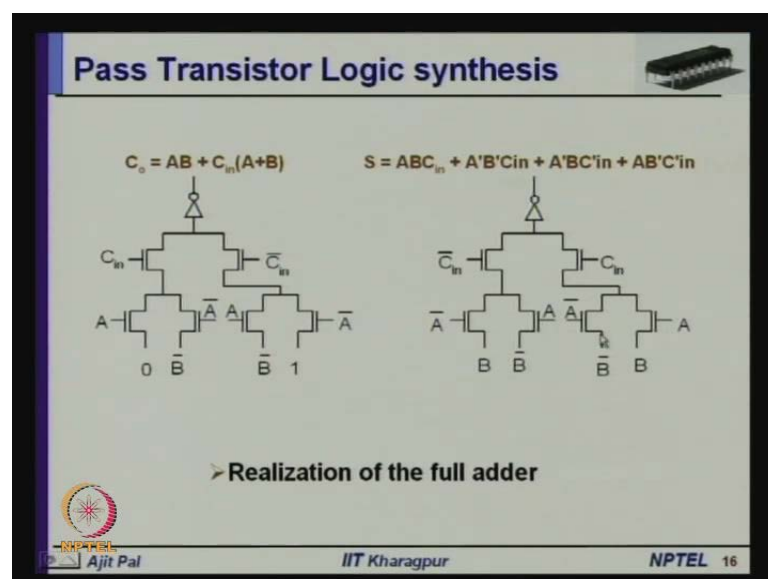
Only necessary nodes and edges are present and this is only for a particular ordering a, b and c. If you take different ordering say start with b then a then c, this particular tree will be different and after this is done, you can do the mapping.

(Refer Slide Time: 54:19)



So this BDD, ROBDD is obtained, then you will do the mapping to the pass-transistor logic tree. Here the mapping has been done. Each of the nodes is mapped with two to one multiplexer. So, here you have used a two to one multiplexer for this node where the control signal is a. This b has been again replaced by two to one multiplexer and c has been node replaced by another two to one multiplexer and we are coming to 0 and 1. So, you can see there is a one to one corresponding between this ROBDD and this is pass pass-transistor tree. Each node has been replaced by a two to one multiplexer. So, this is how you can realize the function.

(Refer Slide Time: 55:02)



Let me illustrate with another example. The example of full adder, you can see here you can perform Shannon expansion or you can get ROBDD for sum and carry functions. And after getting the pass-transistor, getting the BDD or Shannon expansion you can map it to the one of the logic families. Here we have mapped it to that leap cells. So, here you can see this is the y three cell; y three cells are nothing but a fourth to one multiplexer. So, the carry and sum both can be realized by using a, using each by each, for each you require one y three cell; that is your, I mean one cell for each of them. So, this is the realization of each order and you can see the number of transistors required is significantly less. So, we can realize very easily in this particular case.

So with this, we have come to the end of today's lecture. So, we have introduced to you various members of the pass-transistor logic family. And we also have discussed quickly the logic synthesis technique that you can do for pass-transistor logic circuit realization and illustrated with the help of an example of polar diagram. Thank you.