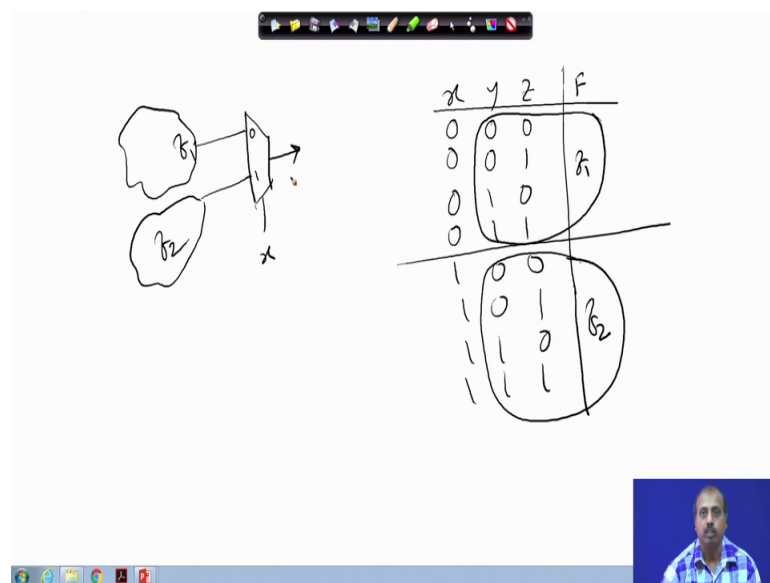


**Digital Circuits**  
**Prof. Santanu Chattopadhyay**  
**Department of Electronics and Electrical Communication Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 26**  
**Decoders, Multiplexers, PLA**  
**(Contd.)**

In our last class we have seen how to realize multiplexer based circuits and we have seen that multiplexers they act as universal gate. So you can realize any of these logic gates using multiplexers. However, the technique that we have seen for multiplexer based circuit realization is that, we start with the truth table of the function and then we partition the truth table so in terms of the select variables.

(Refer Slide Time: 00:42)



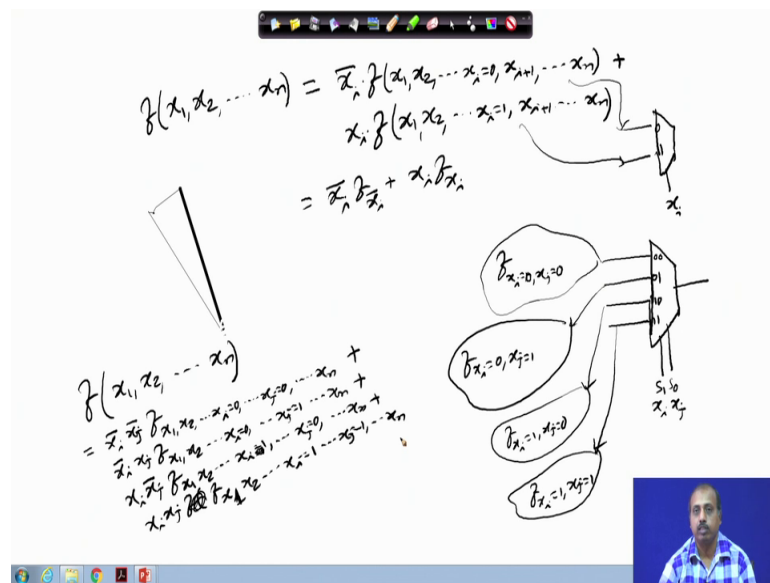
So, what I mean is that if I have got a 3 variable function say a with the variables x y and z and I need to realize it in terms of in terms of a 2 to 1, 2 is to 1 multiplexer then I am doing a partitioning, where this say I am doing a partitioning where this these are all the combination that I have. This is the truth table combination and what we did is we if I have a 2 to 1 multiplexer here then this if x is fed as the control input, then we divide this truth table in to 2 parts ok.

So, x equal to 0 and x equal to 1, for x equal to 0 we realize the function given by this and for x equal to 1 we realize the functions. So if this function is say f 1 and this

function is say f 2 ok. So we realize the 2 functions in f 1 and f 2 and this is the sum we put some combinational gates to realize this f 1 and f 2 and that gives the output.

Now, the problem with this type of technique is that, if the truth table is very large and that is the number of variables are a bit high say more than say 5 or 6 then it is very cumbersome to write down the full truth table and we are partitioning the in this way. A better approach for doing this is via the Shannon decomposition that we have seen and in our last class we have seen that Shannon decomposition around a single variable.

(Refer Slide Time: 02:20)



So, if I have got a say and you have got an n variable function f 1 if f of x 1 x 2 up to x n so this can be decomposed around variable x i as x i bar into f with the function obtained by putting x i equal to 0. And rest of the variables remain as it is plus x i into x 1 x 2 with x i equal to 1 x i plus 1 up to x n. So, this is Shannon decomposition around a single variable.

Now, this is suitable if we are thinking of realizing the circuit in terms of 2 to 1 multiplexers, because this, the input x i will be fed as control input to this stage and the function so they will be getting for 0 and 1. For 0 parts so this will be realized by this function and 1 will be realized by this function, so in short so this is written as x i bar f x i bar plus x i and f x i.

So, this is the notation so  $f_{x_i}$  means, the function obtained from  $f$  by putting  $x_i$  equal to 0 and  $f_{x_i}$  means the function obtained from  $f$  by putting  $x_i$  equal to 1 and these we have functions  $f_{x_i}$  and  $f_{x_i}$ , so they are of  $n - 1$  variables. Now, how can we generalize the method so if we are if we have got a 4 to 1 multiplexer instead of 2 to 1 multiplexer if I have a 4 to 1 multiplexer, that is there are 2 select lines and we have got 4 input lines. So this is for 00, this is for 01, this is for 10 and this is for 11 and these are my select lines say  $S_1$  and  $S_0$ .

So, if I apply say some variable so  $x_i$  here and  $x_j$  there  $x_i$  to  $S_1$  and  $x_j$  to  $S_0$  then for at this part I will look for the function obtained by putting  $f$  of  $x_i$  equal to 1 say  $x_i$  equal to 0 and  $x_j$  equal to 0. Similarly, here I will look for the function obtained by putting  $f_{x_i}$  equal to 0 and  $x_j$  equal to 1, here I look for the function  $f_{x_i}$  equal to 1  $x_j$  equal to 0 and here I will put the function  $f_{x_i}$  equal to 1 and  $x_j$  equal to 1. That is I am looking for decomposition around two variables, so if I write in terms of the similar expression that I have here so we can say do it like this. So we can do it like this we can say  $f_{x_1, x_2, \dots, x_n}$ .

So, if I am decomposing around the variables  $x_i$  and  $x_j$  in this can be written as  $x_i \bar{x}_j$  and  $f$  the function obtained by putting  $x_i$  equal to 0  $x_j$  equal to 0 and rest of the variables they remain unaltered plus  $x_i \bar{x}_j f_{x_1, x_2, \dots, x_i}$  equal to 0  $x_j$  equal to 1  $x_n$  plus  $x_i x_j \bar{f}$  and  $f_{x_1, x_2, \dots, x_i}$  equal to 1  $x_j$  equal to 0  $x_n$  plus  $x_i x_j f$  obtained by putting  $x_i$  equal to 1  $x_j$  equal to 1  $x_n$ . So, this way we can decompose function around any number of variables and accordingly we can get the functions to be sub functions to be realized in different inputs of the multiplexer, so we can take an example.

(Refer Slide Time: 07:33)

Handwritten mathematical derivation and logic circuit diagram:

$$f(a,b,c,d) = ab + \bar{a}cd$$

$$= \bar{a}\bar{b}f_{\bar{a}\bar{b}} + \bar{a}b f_{\bar{a}b} + a\bar{b} f_{a\bar{b}} + ab f_{ab}$$

$$= \bar{a}\bar{b}(cd) + \bar{a}b(c\bar{d}) + a\bar{b}(\bar{c}d) + ab(c\bar{d})$$

The circuit diagram shows a 4-to-1 multiplexer with select lines  $a$  and  $b$ . The data inputs are  $c$  and  $\bar{d}$ . The output is labeled  $f$ .

Suppose I have got a 4 variable function  $f$  a, b, c, d equal to  $a\bar{b}$  or  $\bar{a}cd$  fine. So if we want to realize it using say 4 to 1 multiplexer then we have to first select the choose the select variables, so let us say that we will be using a b as the select variable. So, if we do that a b as the select variable then I have to decompose around a b, so I have to write it as  $\bar{a}\bar{b}f_{\bar{a}\bar{b}}$  plus  $\bar{a}b f_{\bar{a}b}$  plus  $a\bar{b} f_{a\bar{b}}$  plus  $ab f_{ab}$ . Where by  $f_{\bar{a}\bar{b}}$  I mean the function obtained from  $f$  by putting a equal to 0 and b equal to 0 in  $f$   $f_{\bar{a}b}$  means the function obtained from  $f$  by putting a equal to 0 b equal to 1 like that.

So if I just expand it so this becomes  $\bar{a}\bar{b}cd$  so if I in the original function if I put a equal to 0 b equal to 0 so you get  $cd$  fine. Get the functions  $cd$  then  $\bar{a}\bar{b}$  into  $f_{\bar{a}\bar{b}}$  if I put a equal to 0 and b equal to 1, so the first term vanishes in the second term it becomes  $cd$  sorry the second term becomes  $c\bar{d}$  again, because I have if I put a equal to 0 then these term vanishes and this is this becomes 1 and  $cd$  is there so this will also becomes  $cd$  plus  $\bar{a}b$  into  $f_{\bar{a}b}$ .

So,  $\bar{a}b$  so first term again vanishes and the second term  $\bar{a}cd$  since a equal to 1, so this term also vanishes. So this is 0 plus  $ab$  into so this is  $ab$  where this term evaluates to 1 so this is 1. So my final function becomes  $\bar{a}\bar{b}cd + \bar{a}b c\bar{d} + a\bar{b} \bar{c}d + ab c\bar{d}$ , so if I say that I have got a 4 to 1 multiplexer then this is my  $f$  and a and b are fed as input to the select lines then the data line input so this is for 0 0, this is for 0 1, this is 1 0 and this is 1 1

fine. So, here I have to put an and gate with the input c and d the input c and d then 0 1, so it also has got c d as input so you can directly feed it here also then a b bar is 0 so you can put it as 0 here and a b into 1, so you can put a 1 here. So this gives me the realization in terms of 4 is to 1 multiplexer so without you going into truth table.

So, you can use this generalized Shannon expansion and go to the truth table realize, go to the multiplexer based realization of the circuit. So, as the number of select inputs increased, so you have to the decomposition becomes a bit complex and lengthy. So, the decomposition has to be for all the input permutation all the input combinations of the select variables ok. So, with that we will be going to a new topic which is demultiplexer.

(Refer Slide Time: 11:36)

**DEMULTIPLEXERS**

A DEMULTIPLEXER (DEMUX) basically **reverses the multiplexing function**. It takes data from one line and distributes them to a given number of output lines. For this reason, the demultiplexers is also known as a data distributor.

A multiplexer takes **several inputs** and **transmits one of them to the output**.

A demultiplexer (DEMUX) performs the reverse operation ; it takes a single input and distributes it over several outputs.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

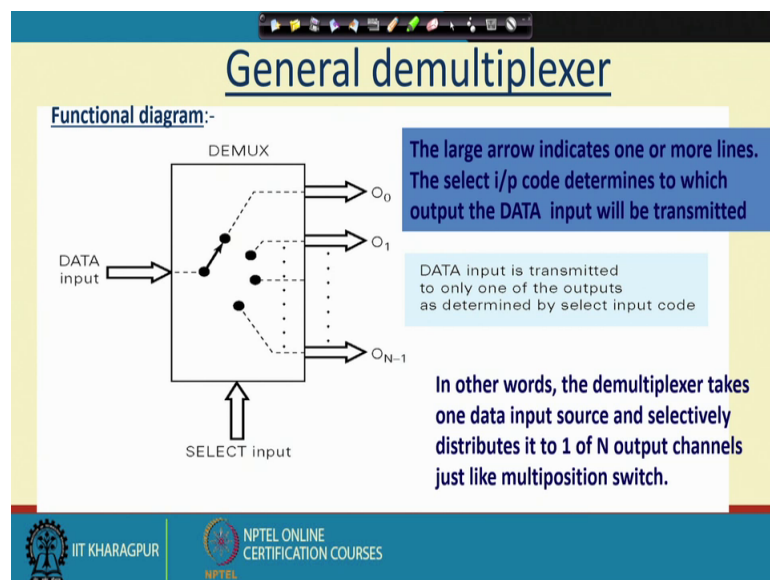
So, this demultiplexer is just the opposite of multiplexer, so in a demultiplexer so we have got some it reverses the multiplexing function, takes data from one line and distributes them to a given number of output lines. So, to which output line the input will go so that is determined by the select input. So, demultiplexer is also known as data distributor as if the data is coming to the block and it is getting distributed to one of its outputs. A multiplexer on the other hand it take several inputs and transmits one of them to the output, so multiplexer is just the demultiplexer is just the reverse of multiplexer and symbolically also you remember that a multiplexer we are showing it like this.

So, we are a multiplexer we are showing it like this, these are the data input so this is the output and so this is the select line so this is the way we are showing the multiplexer.

Similarly for a demultiplexer we will show it like this the data input there is a single data input and there are a number of data outputs. So if I have got a single select line then there will be 2 data outputs.

So if the select line is 0 then the output will be going there, if the select line is 1 then the input will be going to this output. If it is 0 it will go to the upper the input will go to the upper output if it is equal to 1 then it will go to the lower output. Now, a multiplexer so a demultiplexer performs the reverse operation it takes a single input and distributes it over several outputs.

(Refer Slide Time: 13:20)

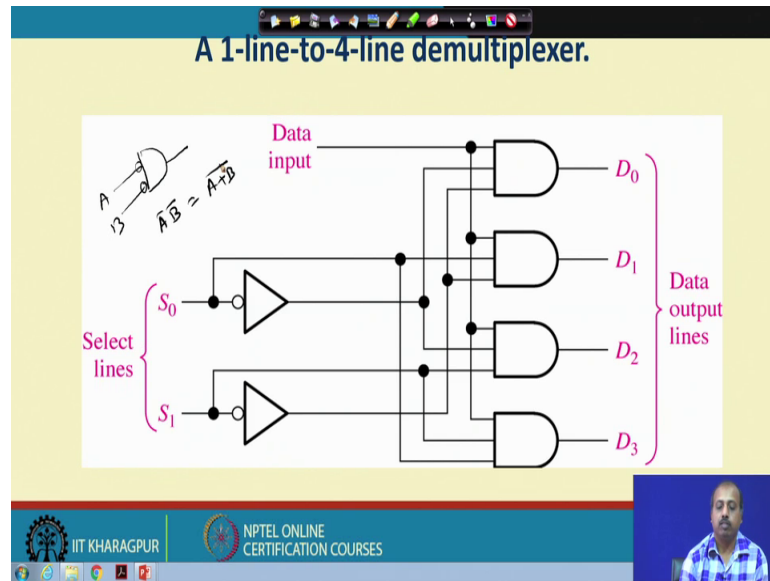


So, then also you can look at it as if there is a switch that goes from that can be get attached to one of the input positions so this to transfer the data input to that line. So here this data input is coming and depending upon the select input so this switch will be set to one of these positions accordingly the input will go to one of the output, so this large arrow indicates one or more lines.

So, it may be the case that instead of having a single line so we have got a multiple line. The select input output code determines to which output the data input should be transmitted. So, it may so happen that instead of getting a single bit input so from it is coming from some block that has got say 16 bit output and the 16 bit output is connected to this demultiplexer as to the input of this demultiplexer to be transferred to a one of many such 16 bit outputs so that way it so this arrow.

So, that may not represent a single line there may be multiple lines also, so data an input is transmitted to only one of the outputs as determined by the select input code. So, multiplexer takes demultiplexer takes one input data source and selectively distributes it to one of n output channels just like the multi positions switch, there is a multi position switch so it is doing it like that.

(Refer Slide Time: 14:46)



So, as far as the logic level diagram is concerned so this diagram shows a 1 line to 4 line demultiplexer or 1 is to 4 demultiplexer. So multiplexer we are writing like 4 line to 1 line or 4 is to 1 multiplexer, for demultiplexer we will write as 1 line to 4 line or 1 is to 4 demultiplexer.

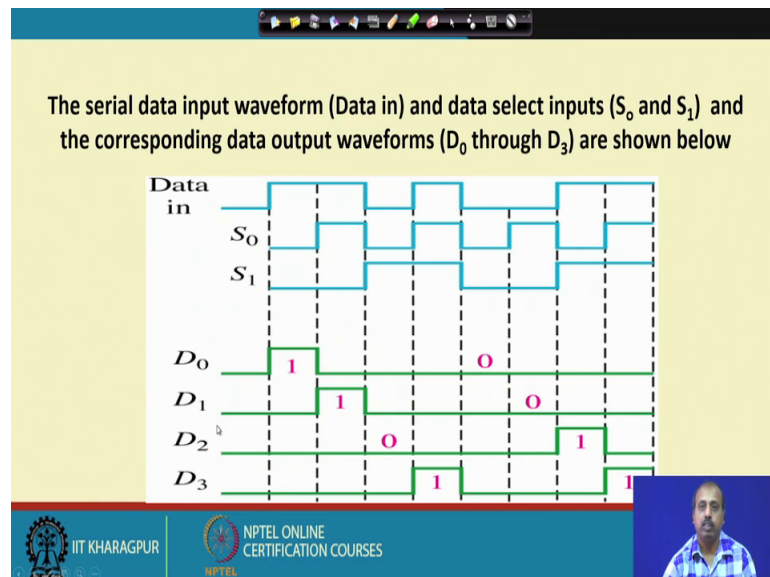
So, here we have got the select lines S 0 and S 1, so if say S 0 S 1 both are 0 0 then what will happen see the this is 0 and this is also 0. So as a result all these 3 and gate output so they will be 0 fine, only so this d 0 so it gets the input from this inverter and it also gets the other input also it gets from the from the output of this inverter.

So, that way it they are coming as 1 and these data input is coming, so whatever be the data input that will pass on to d 0 where as d 1 d 2 d 3 they will get 0s. Now, this multi some so here you get a new notation, so this bubble of the inverter so it is shown before the inverter; So that is also quite common so it does not matter, so it as if the input is inverted here and then buffer or if the inverter is at the output; that means, the input is

buffered and then inverted. So that way it is same previously also we have seen that these bubbles can be put before the gate or after the gate.

So, you have to go by the logic and say what is the function realized for example, if I have got a gate like this and here I have got bubbles then I know that the function realized is and of A bar B bar and of A bar B bar which is nothing, but A or B whole bar, so that is a nor gate ok. So, by applying Demergers theorem we can get it so this way we have to look into the schematic that we have the logic symbols that we have and interpolate what is the corresponding signal.

(Refer Slide Time: 16:56)



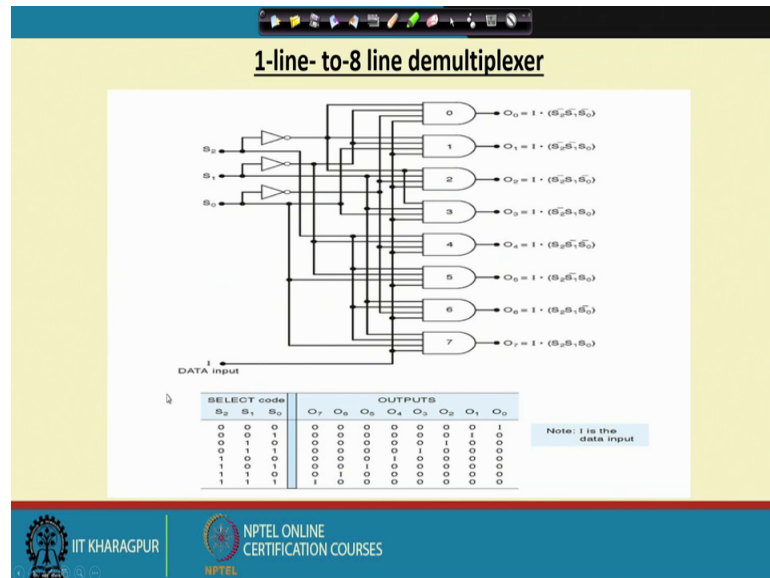
Typical, timing diagram may be like this so this is the serial so data input suppose I have got a data input like this and this  $S_0$   $S_1$  so they are trans doing some transmit they are going like in this fashion. So,  $S_0$  is toggling every in every such boundaries at these dotted boundaries and  $S_1$  is toggling every alternate such dotted boundary. So, initially both are 0 so  $D_0$  will be getting the value of data in and others will all be 0 irrespective of the value of data in after sometime.

So, this  $S_0$  becomes 0 and  $S_1$   $S_0$  becomes 1 and  $S_1$  remains 0, so as a result  $D_1$  get selected so  $D_1$  get this data input at its output and others become all others becomes 0. So in this way you see that at any point of time only one of the outputs will get the value of the data in with it so others will be always others are all 0.



Now, in the of course, we can say that it does not distinguish between 0 which is a coming from the data input and a 0 which is occurring due to non selection of the particular output. So, we will see later that we can have some concept of tri state buffer by which we can get rid of this situation and we can come to a new logic level which is neither 0 nor 1, so that will see later.

(Refer Slide Time: 18:25)



So 1 1-line-to-8 line demultiplexer so that previously what we have seen is a 2 to 4 demultiplexer 2 1 line to 4 line de multiplexer. So, this is 1-line-to-8 line demultiplexer, so naturally I need 3 select lines S 0 S 1 and S 2 and then we have got a number of and gates and this symbols are ended like say this is this is this output is 1. So the here I do not have any data input to be transferred.

So you only say that if it is which one is so this sorry the data input is there and data input is transferred to O 0 if this S 0 S 1 S 2 all are 0s, in that case the data input goes there. If it is equal to S 0 S 1 S 0 is 1 and S 1 S 2 are 0s then this O 1 will get the value of data inputs. So this way this logic circuitry has been made accordingly in the truth table you can see that this I can the input I can pass can come to only 1 of the outputs O 0 to O 7 at any point of time depending upon the select lines S 0 S 1 S 2.

(Refer Slide Time: 19:38)

The slide is titled "Buffers" and contains the following content:

- Buffer:
  - Doesn't change the input.
  - Only amplifies.

There are two diagrams illustrating buffers. The first is a simple buffer symbol with an input line on the left and an output line on the right. The second is a more detailed buffer symbol with an input line labeled "in", an output line labeled "out", and an enable line labeled "EN" on the top.

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a man is visible in the bottom right corner.

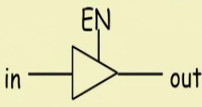
Next we will be looking into another fundamental concept which is known as buffer, so what happens is that may when whenever we are transmitting signals over long distance so due to this noise and this RC drop of the line. So the voltage level of the signal drops so maybe you have transmitted a logic high, but due to the length of the wire at the other end when you measure, so you do not get a strong logic high so you get some intermediary value or in the worst case it may even become a logic low level. So, what is required is that we should do some amplification of the signal so such signal voltage level has to be restored to the proper values. So, if I have a line like this so if I have a line like this.

So, what I means so at this point I am sending some value signal S at this point when I measuring s I find that it is not that strong so what is done in between we put some buffer like this buffer as it is shown here. So, this buffer will amplify the signal so it does not so it is basically a true value transfer, so whatever is coming as input will go to the output provided this enable line is equal to 1, if this enable is equal to 1 then in output equal to input. So it does not do any other logic change to the signal in so.

(Refer Slide Time: 21:13)

### Three-State Buffers


- Buffer output has 3 states: 0, 1, Z
- Z stands for High-Impedance  $\cong$  Open circuit



EN	in	out
0	X	Z
1	0	0
1	1	1

EN = 0  $\rightarrow$  out = Z (open circuit)  
EN = 1  $\rightarrow$  out = in (regular buffer)

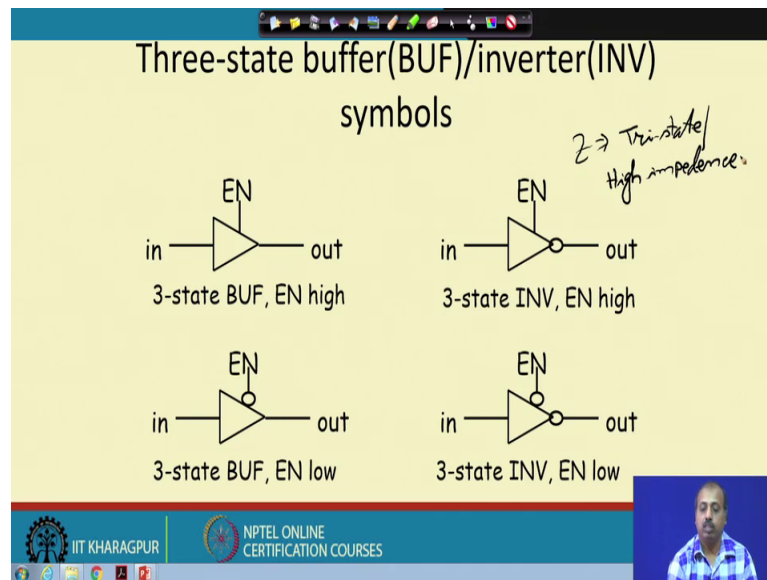
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, these are also known as 3-state buffers because, when this what happens when enable equal to 0 when enable equal to 1 then I have said that output gets input, so what if output sorry enable line is equal to 0. So, this type of buffers where we have got an enable line so they have got 3 output states 0 1 and Z.

So, Z is Z is known as the high impedance state or open circuit state, so as a as if the circuit is open it is not connected to any voltage source. So, enable equal to 0 output equal to Z which is equivalent to the open circuit and enable equal to 1 output is output is equal to in that is the regular buffer operation that will occur there. So, enable equal to if we if we do not want the output to get the value of input.

(Refer Slide Time: 22:14)



So, we can put enable equal to 0 and that will stop the output getting the value of input. Now, there are many such 3-state buffer configurations so is so I can have these buffer with enable high, enable high means when this enable equal to 1 then only this output will get the value input. Sometimes we have got a so this is a buffer though this is a 3-state inverter so this is a basic inverter logic, but before that we have got so we there is an enable line also so if the enable is high then only this circuit will act as an inverter otherwise in this output will be tri stated.

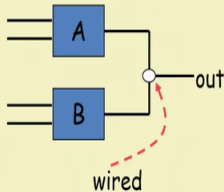
So, output is the output will get the value  $Z$  so this  $Z$  is also known as that tri state  $Z$  that I am talking about so that is also called the tri state. There are several names one name we have already seen the high impedance state, the high impedance state or the tri states state like that we have got several names for this.

Now, this 3 so this so we have got this thing we have got this enable and if enable is 1 so this is a basically an inverter with some enable line. This enable can be the enable itself may be an inverted enable that is this buffer will get the value of output; output will get the value of in provided this enable line is equal to 0. If enable line is low then output gets the value of input and if these enable line it is similarly I can have an inverter where the enable line itself is low. So, this is we have got this enable equal to 0 then only this circuit acts as an inverter, so that is possible. So, we can have various combinations of this buffer.

(Refer Slide Time: 24:05)

### Open Collector/Drain Gates

– Outputs of two (or more) gates must not be wired together:



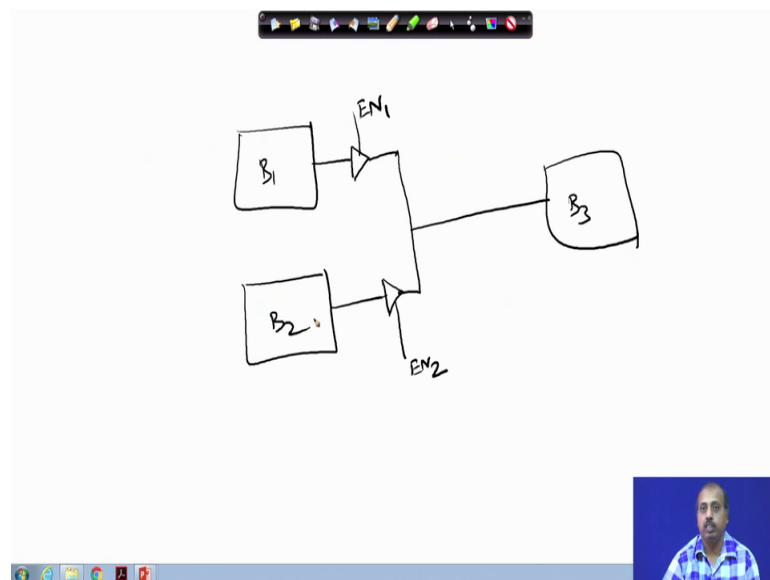
If  $A = B$   
→  $out = A = B$

If  $A \neq B$   
→ a large enough current can be created, that causes excessive heating and could damage the circuit

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Next we will look into let us let us see like why this is necessary? So this is necessary because many times what happens is that we have got different logic blocks they are giving me some sources.

(Refer Slide Time: 24:26)



So, this is see this is say block 1 whatever it is, so this is block 1 this is block 2 so they are all logic signal they are all logic blocks computing some logic operation and then we need to give this output of both the blocks to block B 3. So block B 3 so this output should be this input should get it should be coming from the output of B 1 or B 2, but at

any point of time only 1 of them has to be connected and sometimes this input may not be connected also what I mean is that sometimes this B 3 gets the value from B 1 or B 2 or it may be that it is not connected at all.

Now, we will see later that if I do it like this if I connect it in this fashion, so what I really want is something like this that these are the outputs from B 1 and B 2 and this is going there to B 3. Now, this type of connection is difficult because what may happen is that there may be conflict in the outputs of B 1 and B 2 and in that case so what is the voltage level at this point so this becomes a question fine.

So to avoid that situation so we can use the tri state gates, the high impedance gate so what we do is we put buffer here sorry so let us take a new page. So, we take we take we have got this B 1 and output of it is put to a buffer and we have got a block B 2 and this output of B 2 we passes through another buffer and this two are now connected and that goes to the block B 3 fine. Now, this enable line so they will be so this is say enable 1 and this is enable 2.

So, if enable 1 so this B 1 output will be coming to the input of B 3 if enable 1 is high and likewise a B 2 will B 2 output will come if enable 2 is high. Now, you see that if enable 1 and enable 2 they are not activated simultaneously then I do not have any problem. So, this B 1 output is never getting connected to shorted to B 2 output ok, so that will not happen. So, this way we can use this tri state logic for realizing for realizing this type of connections where multiple outputs from different blocks they may be connected to a common input of another block.