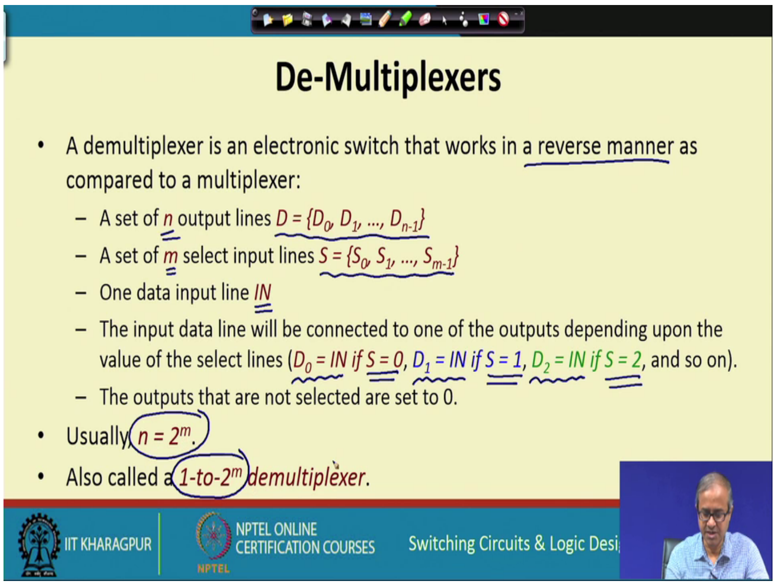
Switching Circuits and Logic Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institution of Technology, Kharagpur

Lecture – 25
Logic Design (Part - II)

So, we continue with our discussion on Logic Design. This is the second part of the lecture logic design part 2; now you recall in our earlier lecture we talked about multiplexers. So, continuing with this discussion we talk about another kind of a basic building block, which is similar to a multiplexer, but it can be considered as its inverse or reveres. This is called a de-multiplexer, let us start with that a de-multiplexer.

(Refer Slide Time: 00:48)


De-Multiplexers

- A demultiplexer is an electronic switch that works in a reverse manner as compared to a multiplexer:
  - A set of n output lines $D = \{D_0, D_1, \ldots, D_{n-1}\}$
  - A set of m select input lines $S = \{S_0, S_1, \ldots, S_{m-1}\}$
  - One data input line IN
  - The input data line will be connected to one of the outputs depending upon the value of the select lines ($D_0 = IN$ if $S = 0$, $D_1 = IN$ if $S = 1$, $D_2 = IN$ if $S = 2$, and so on).
  - The outputs that are not selected are set to 0.
- Usually $n = 2^m$.
- Also called a $1\text{-to-}2^m$ *demultiplexer*.

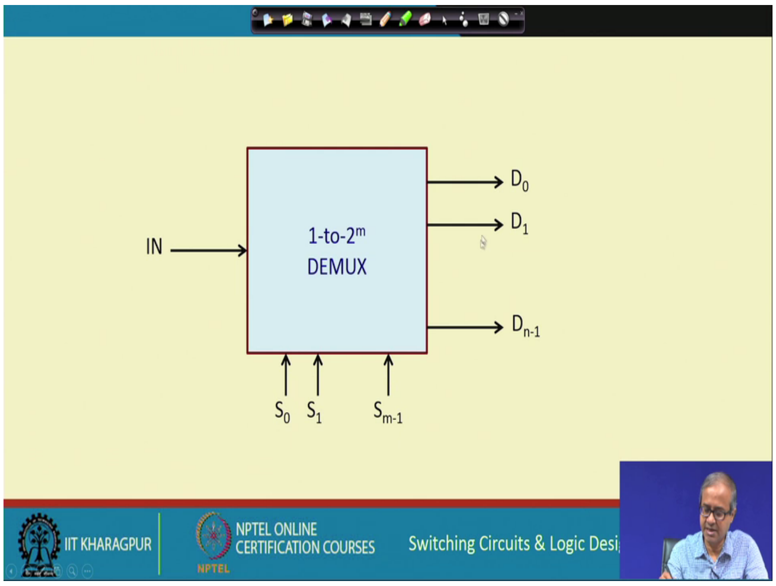
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, why we say this is an inverse? This works in a reverse manner as compared to a multiplexer. You see for a multiplexer there are many inputs and one output, many inputs and output depending on the select line one of the inputs is selected. But in a de-multiplexer there will be one input, but many outputs depending on the select line the input will be connected to one of the outputs, this is why we call it the reverse or inverse.

So, a de-multiplexer will be having n number of output line that is called D 0 to D n minus 1. So, just like a multiplexer there will be m number of input lines where n equal to 2 to the power m and there will be one data input line here let us say IN. So, the way the de multiplexer will function is very similar to multiplexer just the reverse of it when

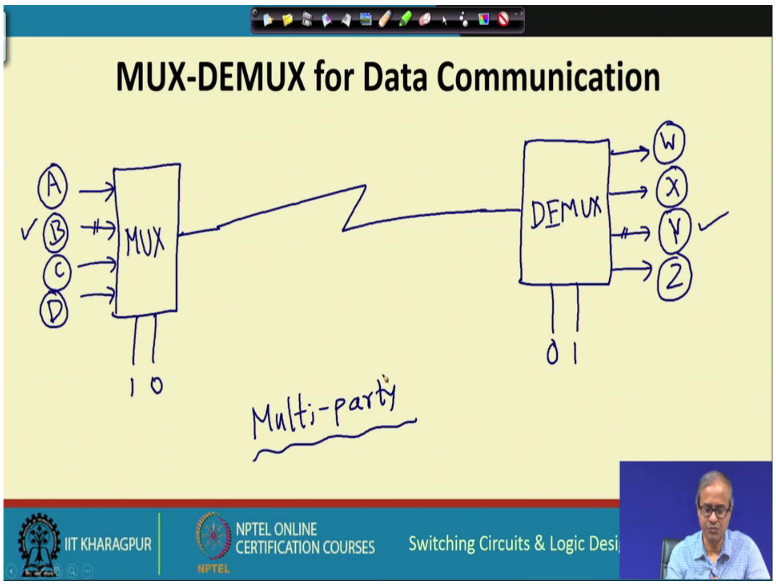
S is 0; that means, it is all 0 IN will be connected to D 0. If S is 1 IN will be connected to D 1, if S is 2 IN will be connected to D 2 and so, on right. This is called a 1 to 2 to the power m because there are 2 to the power m outputs n is 2 to the power m de multiplexer ok.

(Refer Slide Time: 02:34)



So, as I had said a de-multiplexer looks like this, there is one input n number of outputs and select lines, which will be selecting one of the outputs the input will be connected to that. Now, you may think where we can or what are the applications of this de-multiplexers and so, on; there will be many applications in fact, but one application I would like to just highlight their use in data communication.

(Refer Slide Time: 02:55)


MUX-DEMUX for Data Communication

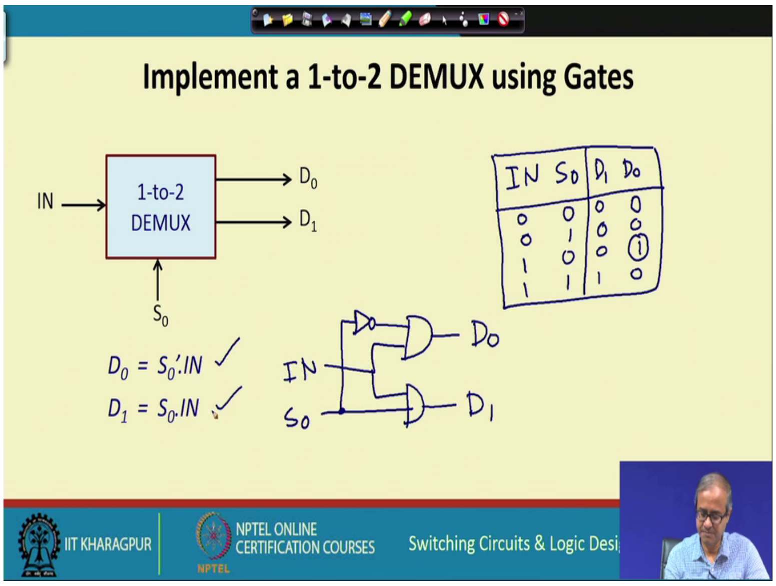
A
B
C
D
MUX
1 0
DEMUX
0 1
W
X
Y
Z
Multi-party
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Let us consider a scenario, suppose I have a multiplexer here let us consider 4 input multiplexer and on the other side let us say I have a de-multiplexer, let us say here also there are 4 outputs. So, multiplexer is having one in one output DEMUX have been one input and suppose they are connected let us say over a long distance connection line they are connected.

So, MUX will have two select lines DEMUX will also 2 select lines. Let us say this inputs are connected to A B C D some input sources and this outputs of the DEMUX's are connected let us say W X Y Z. Now, you imagine that this is something like a telephone exchange, you think of the older telephone exchange which you had many years back, we are used to dial the numbers and using some switches in the exchanges the connection used to get established.

So, conceptually this is something like that suppose A B C D are subscribers W X Y Z are also subscribers let us see B is trying to make a call to Y. So, what the telephone exchange will do the telephone exchange will be applying 0 1 to the select line of here. So, that B get selected and it will apply 1 0 to the select line here. So, that this output gets selected. So, ultimately this B will be available on this line and this will get connected to Y. So, for multiparty communication, this kind of multiplexer de multiplexer pair can be very useful at as this simple example shows it right ok.

(Refer Slide Time: 05:25)



Implement a 1-to-2 DEMUX using Gates

IN
1-to-2 DEMUX
D_0
D_1
S_0
IN
S_0
D_0
D_1
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

$D_0 = S_0'.IN$ ✓

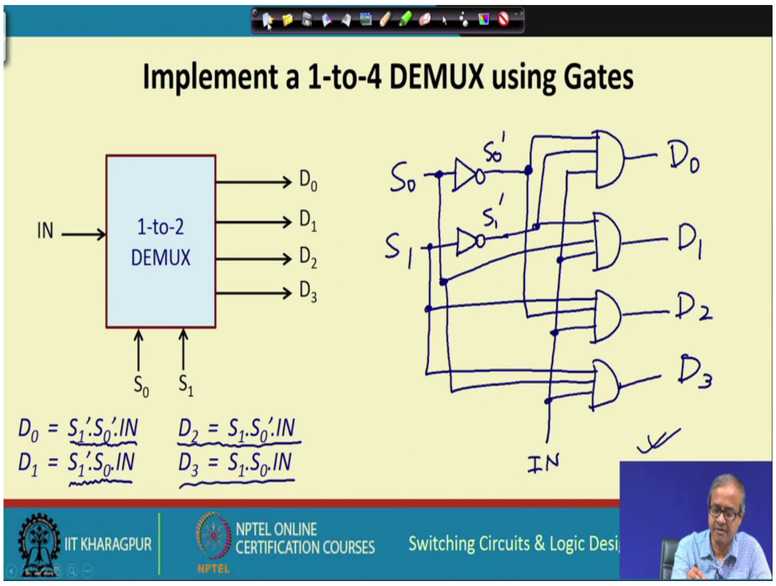
$D_1 = S_0.IN$ ✓

IN	S_0	D_1	D_0
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

Let us move on let us see how you can implement a small de multiplexer there is 1 input and there are 2 outputs. First again let us considered a truth table approach, this is a very simple case there only 2 inputs there is IN there is S 0 and there are 2 outputs D 1 D 0. So, inputs can be 00 0 1 1 0 and 11. Now, if the input is 0 0 select line is 0. So, this 0 will go to D 0 if select line is 1 input will go to D 1, select line is 0 this input 1 will go to D 0 and if it is 1 it will go to D 1, then assuming the output which are not selected they will remain at 0.

So, this is my function. So, if you construct the truth table you can I mean see the this cannot be minimized any further. So, what it D 0 indicate? D 0 has a single 1 here this is S 0 bar into N, let us see D 0 is S 0 bar N and D 1 is S 0 IN. So, if you want to implement it you just need two AND gates, they will be generating D 0 and D 1 and this input IN will be connected to both. And S0 will be connected directly to here and via a NOT gate to here, this will be simple implementation for 1 to 1 MUX using gates right, just implement these two functions just 3 gates you require fine.

(Refer Slide Time: 07:30)



Implement a 1-to-4 DEMUX using Gates

$D_0 = S_1'.S_0'.IN$ $D_2 = S_1.S_0'.IN$

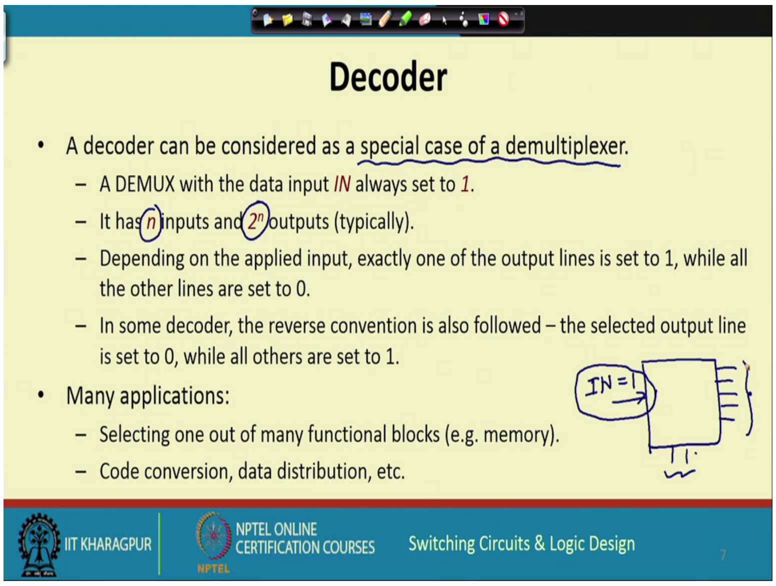
$D_1 = S_1'.S_0.IN$ $D_3 = S_1.S_0.IN$

Now, think of a larger DEMUX it is a 1 to 4 because there are 4 outputs there will be two select lines. Now, again instead of going through the truth table, let us try to find out how we can directly write down the expressions. You see D 0 will be what I am saying in will go to the D 0, if both S 0 S 0 S 0 S 1 0 and 0. So, we write it like this S 1 bar S 0 bar IN and of this 3 D 1 will be S 1 is 0 S 0 is 1, S 1 bar is 0 IN, D 2 is similarly S 1 is 0 bar 1 0 and D 3 is 11.

So, how you can implement this very easily? We would be requiring 4 AND gates to generate the 4 outputs D 0, D 1 D 2 and D 3. Let us say one of the inputs is common to all this is connected common to all of them that is your IN and this S 0 and S 1 which are there, let us say that we have also have some NOT gates connected. So, we have S 0 bar and S 1 bar available with us. This is S 0 bar, this is S 1 bar. So, in the first gate for D 0 I need to connect S 1 bar and S 0 bar. So, I connect S 0 bar I connect S 1 bar, for D 1 I need S 1 bar and S 0. So, I need S 1 bar and S 0 like this for D 2 I need S 1 and S 0 bar.

So, I need S 1 and S 0 bar S 0 bar is this S 0 bar this and D 3, I need S 1 S 0. I connect S 1 I connect S 0, just like this to look at this expression just connect either S 0 and S 0 bar S 1 and S 1 bar accordingly. So, you get a de-multiplexer circuit right quite simple fine.

(Refer Slide Time: 10:20)



Decoder

- A decoder can be considered as a special case of a demultiplexer.
  - A DEMUX with the data input *IN* always set to *1*.
  - It has *n* inputs and 2^n outputs (typically).
  - Depending on the applied input, exactly one of the output lines is set to 1, while all the other lines are set to 0.
  - In some decoder, the reverse convention is also followed – the selected output line is set to 0, while all others are set to 1.
- Many applications:
  - Selecting one out of many functional blocks (e.g. memory).
  - Code conversion, data distribution, etc.

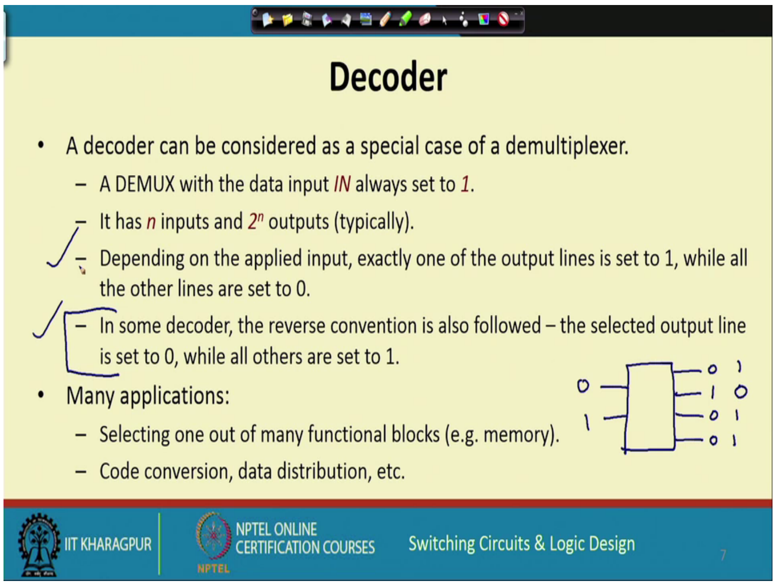
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Now, let us come to a special kind of a de-multiplexer it has a different name it is called a decoder. Decoder you can say is a special type of a de multiplexer, but it has very specific and different kinds of applications. So, what is a decoder? As I had said this is a special case of a de-multiplexer what kind of a special case, you can say that I have a de-multiplexer where there was one input and there was several outputs and select lines. So, I am assuming that the input line is always equal to 1.

So, under that special case I shall be calling this decoder and because I am assuming this to be always 1; so, this input need not be connected explicitly. So, I will say that only my inputs are the select lines and my only output at the just the output lines.

So, a decoder essentially will be having n number of inputs and 2 to the power n number of output let us say right depending on the applied input. So, whatever you are applying exactly one of the output will be set to 1, while the others are set to 0 like let us take an example

(Refer Slide Time: 11:43)



Decoder

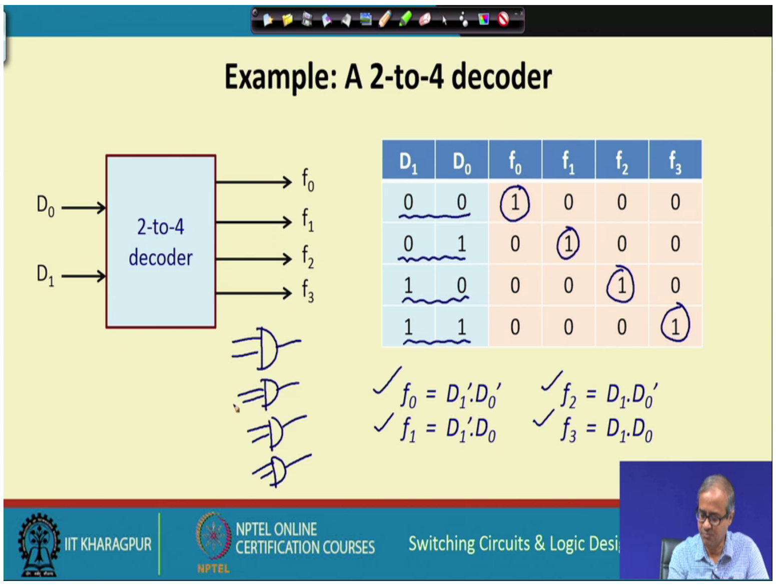
- A decoder can be considered as a special case of a demultiplexer.
  - A DEMUX with the data input *IN* always set to *1*.
  - It has *n* inputs and 2^n outputs (typically).
  - Depending on the applied input, exactly one of the output lines is set to 1, while all the other lines are set to 0.
  - In some decoder, the reverse convention is also followed – the selected output line is set to 0, while all others are set to 1.
- Many applications:
  - Selecting one out of many functional blocks (e.g. memory).
  - Code conversion, data distribution, etc.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Design

Suppose I have a decoder with 2 inputs and 4 outputs let us say I have applied 0 1; 0 1 means 1. So, 0 1 let us say this line will be set to 1 all others line will be set to 0. If I apply with a 11 then the last line will be 1 others will be 0. Now, the point to notice that which you have mentioned here is that, some decoders are available where the reveres convention is followed. So, what is the reverse convention? It says that the line which is selected that will be set to 0 and all others are set to 1.

So, you see design is the same just on the output as if you are connecting a not gate right. So, either you follow this convention or you follow this convention. There are many applications of a decoder like when you have many functional block. For example, memory in a computer system there can be many memory modules depending on the address you have to select one of the modules you can use a decoder to select that. So, there were many applications were not talking about them right now, some of the application we shall be seeing later possibly, but not right now. There are other applications like code conversions data distribution and many others.

(Refer Slide Time: 13:20)



Example: A 2-to-4 decoder

D_0
D_1
2-to-4 decoder
f_0
f_1
f_2
f_3

D_1	D_0	f_0	f_1	f_2	f_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

✓ $f_0 = D_1'.D_0'$ ✓ $f_2 = D_1.D_0'$

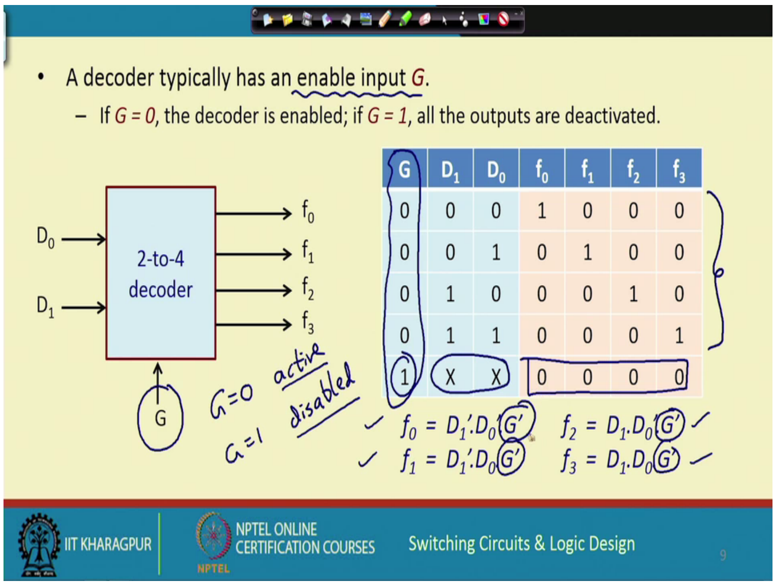
✓ $f_1 = D_1'.D_0$ ✓ $f_3 = D_1.D_0$

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Let us take a specific case 2 to 4 decoder. So, you have a 2 to 4 decoder whether a 2 inputs 4 outputs look at the truth table. Depending on the data info let us say if it is 0 0 and f 0 is set to 1 for let us say I am following this convention the others are 0 the line selected is 1. If it is 0 1 f 1 is selected, 1 0 f 2 is selected and 11 f 3 selected.

Now, here also you can straight way write down the expressions for the 4 outputs you see f 0 will be 1 only when D 1 D 0 at 0 0. So, you can straight away right f 0 is equal to D 1 bar D D 0 bar. Similarly, f 1 will be 1 when it is 0 1 D 1 bar D 0, f 2 1 0 D 1 D 0 bar and finally, f 3 1 1 D 1 D 0. So, again just to implement decoder you need 4 AND gates all of them will be 2 inputs; this will be generating the 4 outputs f 0 f 1 f 2 f 3 and the inputs you apply accordingly, D 0 D 1 are there you will be requiring 2 NOT gates you can apply the inputs accordingly right ok.

(Refer Slide Time: 14:56)



- A decoder typically has an enable input *G*.
  - If *G = 0*, the decoder is enabled; if *G = 1*, all the outputs are deactivated.

G	D_1	D_0	f_0	f_1	f_2	f_3
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	X	X	0	0	0	0

$f_0 = D_1'.D_0'G'$ $\quad f_2 = D_1.D_0'G'$

$f_1 = D_1'.D_0G'$ $\quad f_3 = D_1.D_0G'$

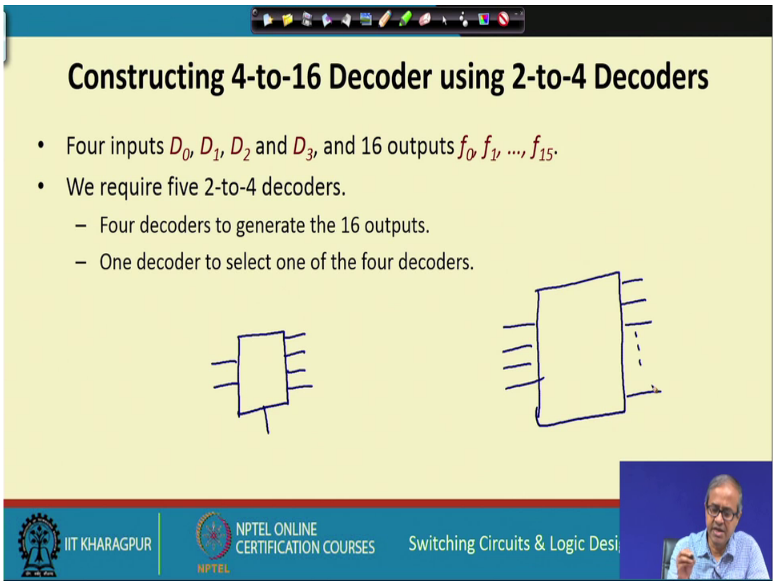
Now, the point to notice that the decoders that are available, the we use they typically has another input called enable like normally a decoder has 2 inputs and 4 outputs like a 2 to the 4 decoder, there will be another input called enable using which I can disable the decoder if I want. The idea is that if a decoder is disabled, then irrespective of what we apply in the input none of the outputs will get selected, only when decoder is enabled then only depending on the input the respective output will gets selected.

Now, this feature you require when you have many decoders, I will take an example later and you need to select one of the decoders. So, you enable one of them you disable all others the concept is like this ok. Now, in this case the truth table will be very similar the only thing is that I have an extra input call G, where the normal operation of the decoder which is shown in the first 4 columns the G 0, but when G is 1.

G 0 means I am assuming that this is active low; that means, G equal to 0 means the decoder is active and G equal to 1 means the decoder is disabled. So, when the decoder G is 1 irrespective of what we apply in the input the outputs will always 0 none of them will be selected right. So, when the expressions for the outputs you will be similar only you add a literal G bar to all of them.

Because they will be 1 if is also 0. So, G bar will get added. So, you will be need in the and gates here which will be 3 inputs each in the same way you can implement right let us take some examples.

(Refer Slide Time: 17:19)



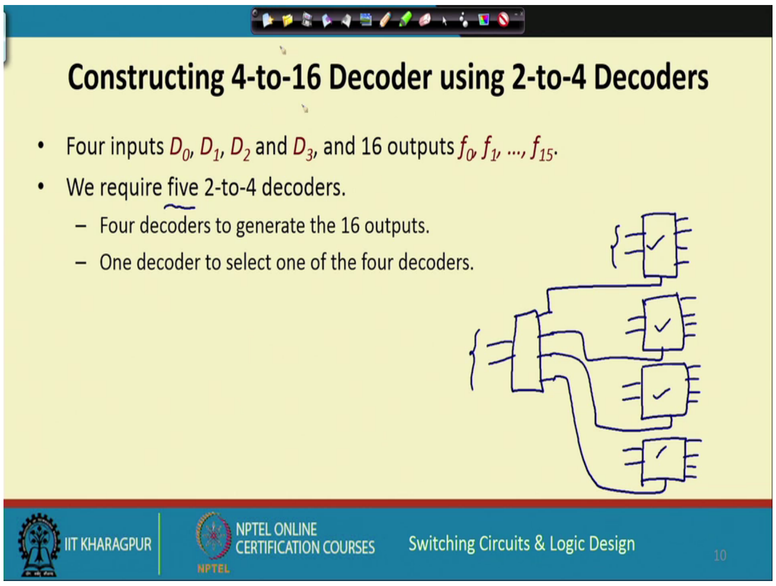
Constructing 4-to-16 Decoder using 2-to-4 Decoders

- Four inputs D_0, D_1, D_2 and D_3, and 16 outputs f_0, f_1, ..., f_{15}.
- We require five 2-to-4 decoders.
  - Four decoders to generate the 16 outputs.
  - One decoder to select one of the four decoders.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Suppose I have a smaller decoder 2 to 4 decoders, these are available to me. So, I have this kind of decoder available to me 2 inputs 4 outputs and enable, but I want to design a larger decoder 4 inputs and 16 outputs. So, how do you do it? Now, we shall be showing you the design the basic idea is like this we will be requiring 5 such smaller decoders.

(Refer Slide Time: 18:00)



Constructing 4-to-16 Decoder using 2-to-4 Decoders

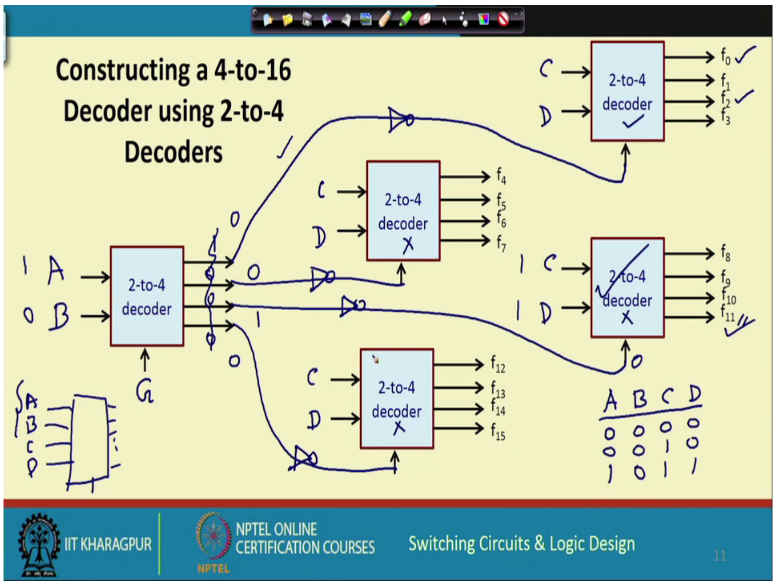
- Four inputs D_0, D_1, D_2 and D_3, and 16 outputs f_0, f_1, ..., f_{15}.
- We require five 2-to-4 decoders.
  - Four decoders to generate the 16 outputs.
  - One decoder to select one of the four decoders.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design
10

In what way the decoders will be connected somehow like this somewhat in this fashion well, I shall be showing you the detailed diagram there are 4 2 to 4 decoders will be 2 inputs 4 output each.

So, this will make 16 outputs and there will be another decoder which will also be a 2 to 4 decoder this decoder actually will be selecting one of these 4 decoders. So, depending on what we apply here, one of these 4 decoders will get selected and depending on what you apply in input of that decoder, the corresponding output will get selected. This is the basic concept regarding the construction of the 4 to 16 decoder here.

(Refer Slide Time: 19:00)


Constructing a 4-to-16 Decoder using 2-to-4 Decoders

Let us see. So, here I have this schematic let me complete the connections. So, what do you want here just a second, let me just make this. There is a formatting issue let me correct this 14, this is 15 this is 10 and this is 11 fine.

So, let us have the so, now, we have a 4 to 16 decoder let us consider how our 4 to 16 decoder will look like our 4 to 16 decoder. Let us say there will be 4 inputs let us call them A B C D, where A is the most significant bit B is the less significant bit. So, let us see how we connect the inputs to the decoder, what we say is that we will be connecting A and B here the most significant 2 bits of the decoder, and C D we connect to the inputs of the other 4 decoders.

Now, you see as I had said in the previous diagram that the 16 outputs are generated by this 4 decoders f 0 to f 3 here, f 4 to f 7 here f 8 to f 11 here and f 12 to f 15 here. And this decoder what it does this will be selecting one of the 4 decoders. This is the first decoder, this is the second decoder, this is the third decoder and this is the fourth decoder and this will be the enable of the whole decoder this is the G of this 4 to 16 decoder.

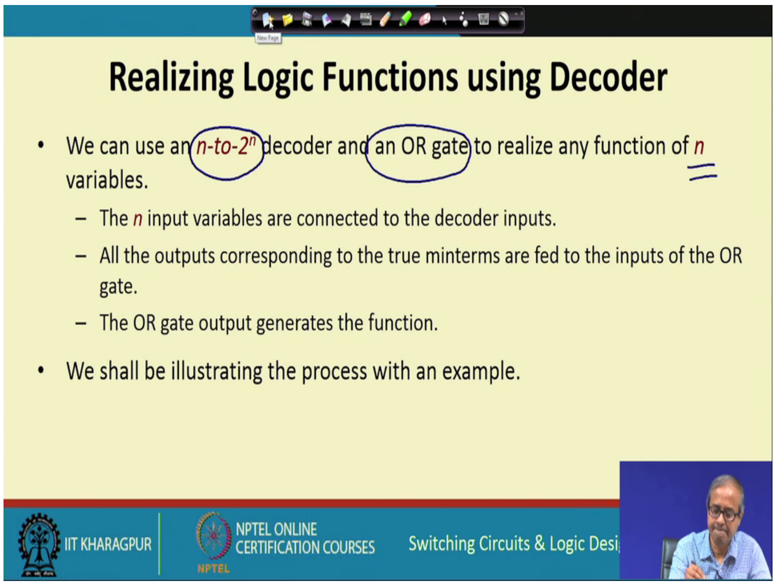
Let us see how this works. So, as I had said this A B C D or the 4 select lines, let us take some examples suppose I have applied 0 0 0 and 0. So, under this condition f 0 is supposed to get selected; that means, f 0 should be one all other should be 0 let us say how it works. Since A B is 0 0, for the first decoder this one only will be one all others will be zeros for the reverse.

So, let us consider I mean if we consider the convention that you are followed then will be connecting a not gates here let us assume there is also not gate in between here they are connected via not gates. So, if this is 1. So, this decoder will be selected, but these are all 0 0 0. So, these enables will be 1 1 1 these are not selected not selected not selected and C D is also 0 0. So, C D is 0 0 means this f 0 will get selected.

Now, let us say I apply 0 0 1 0, because A B is 0 0 again this one will be selected this line, C D is 1 0 means this is the decoder which is selected C D 1 0 means the third line this f 2 will be selected that is correct. Let us take another example suppose I apply 1 0 11. 1011 which is supposed to be selected is 10 and 1, 11 this f 11 is supposed to be selected; let us see how? In A B I have applied 1 0, in A B I have applied 1 0. So, in the output this is not the case because is 1 0 this line will be selected the third line, the others will not be selected. Because the third line is selected this will be 0 and now this decoder will be selected, now C D I have applied 11 11 it will select the last one f 11. So, you will see this works.

So, in this way we can very systematic look and design an arbitrary 4 by 16 decoder using 2 to 4 decoder. As I had said there are many applications where you need to design very large decoders. So, it is important to know how such smaller decoders can be connected we say cascaded together to form larger decoders. So, using this very small example, I think I am able to show you how smaller decoders can be used to construct larger decoders. In this example we had used 2 to 4 decoders to construct a 4 to 16 decoder.

(Refer Slide Time: 24:40)



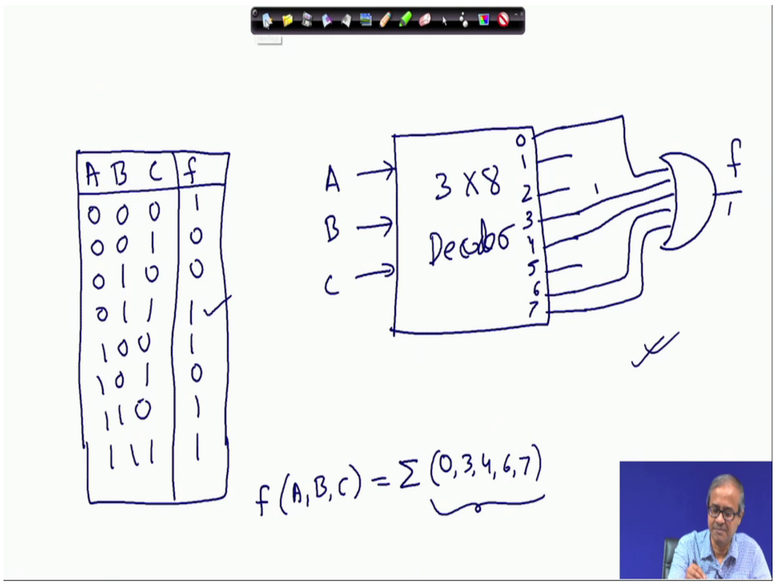
Realizing Logic Functions using Decoder

- We can use an *n-to-2^n* decoder and an OR gate to realize any function of *n* variables.
  - The *n* input variables are connected to the decoder inputs.
  - All the outputs corresponding to the true minterms are fed to the inputs of the OR gate.
  - The OR gate output generates the function.
- We shall be illustrating the process with an example.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, the last thing you consider here is that, how we can implement logic functions can we implement logic functions using decoder also. We say that will yes it may not be very natural, people may not be doing it because multiplexers and much more convenient. But decoders can also be used to implement some logic function implement some truth table let us see how. So, the idea is that you use and n by 2 to the power n decoder and an OR gate to implement a function of n variables.

(Refer Slide Time: 25:37)



A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1 ✓
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$f(A,B,C) = \Sigma \underbrace{(0,3,4,6,7)}$$

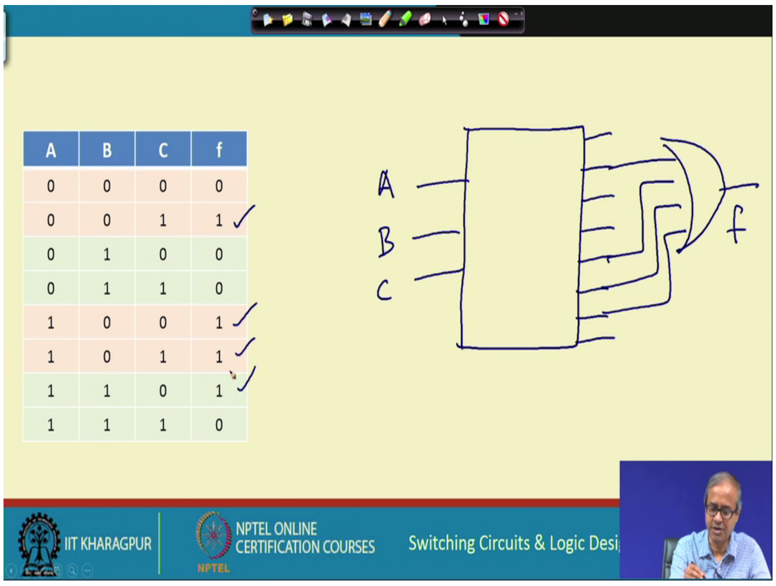
So, how do we do it let us take an example, let us consider an example let us say that I have a truth table; consider a truth table say this is a 3 variable function A B C and f. These are the inputs, let us say the output column is looks like this there are 5 1's in the truth table.

So, in terms of the function representation, you can also write it like this represents the function sigma notation it is 0 3 4 6 7; 0 3 4 6 and 7. Now, how we can implement using decoder? What we say is that we take a 3 to 8 decoder. So, we connect the 3 inputs here A B C there will be 8 outputs this 8 outputs will be corresponding to the different combination. So, A B C to a selected and writing down the numbers 0 1 2 3 4 5 6 and 7.

Now, you look at this what are the minterms that are true minterms 0 3 4 6 7, you connect an OR gate here 0 connect one input 0, 3 connect 3, 4 connect 4 connect 6 connect 7 this will be your output. So, it is easy to see how it works, depending on A B C value if one of the true minterms is applied.

So, the corresponding output will be one let us say you have applied 0 1 1. So, this one will be 1, this an OR gate if any one of the input is 1 the output will be 1. So, in this way you can implement the function right very simply. But you may need a larger or gate with large number of inputs depending on how many ones are there in your function right.

(Refer Slide Time: 28:20)



A	B	C	f
0	0	0	0
0	0	1	1 ✓
0	1	0	0
0	1	1	0
1	0	0	1 ✓
1	0	1	1 ✓
1	1	0	1 ✓
1	1	1	0

A
B
C
f
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, this is another example. So, you can similarly work out here. If you apply A B C, again you have the 8 outputs here the n 2 minterms are there 0 0 1 which is this 1 0 0 4 1 1 2 3 4 this one, 5 and 6 5 and 6 connect them to the OR gate this will be a function ok.

So, these are also very programmable kind of a thing the decoder remains the same only the OR gate and how you connect them that will vary you can implement any arbitrary function alright. So, with this we come to the end of this lecture. Now, we shall be continuing with our discussion in the next lecture, where we shall be talking about some other kind of building block; some special kind of decoders and other building blocks that are also very useful in logic design.

Thank you.