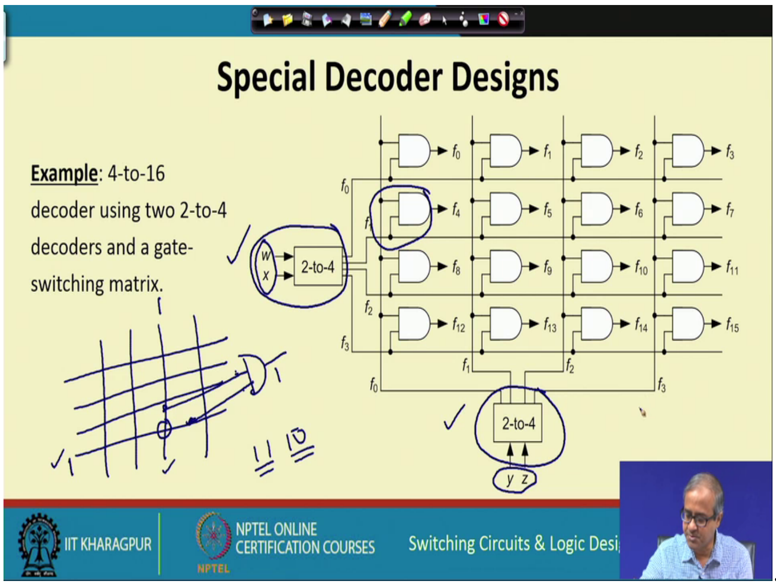
Switching Circuits and Logic Design

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 26
Logic Design (Part III)

So, we continue with our discussion on Logic Design, this is the third part of our logic design lecture.

(Refer Slide Time: 00:30)



Special Decoder Designs

Example: 4-to-16 decoder using two 2-to-4 decoders and a gate-switching matrix.

2-to-4
w
x
y z
f_0 f_1 f_2 f_3
f_4 f_5 f_6 f_7
f_8 f_9 f_{10} f_{11}
f_{12} f_{13} f_{14} f_{15}
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Now here, the first thing that I would like to talk about is to discuss some kind of special decoder design, but before I go into it let me try to give the motivation, see we talked about the design of decoders. Now, you imagine that you are trying to design a 4 to 16 decoder.

Now, if you design 4 to 16 decoder directly using gates not using smaller decoder as you are seen, directly using gates, then for every output you need a large gate you need an and gate with 5 inputs, 4 inputs will be some combination of the input variables and one for the enable. So, you will be needing 16 AND gates each with 5 inputs.

Now one difficulty in designing circuits is that when the number of inputs of a gate increases, it becomes increasingly difficult to design and manufacture those gates while the gates become unreliable, they consume more power they become slower and there are many drawbacks. So, it is always attempted to keep the number of inputs to a gate as smallest possible. So, instead of designing a very large decoder just in one go it is always better to design smaller decoders and use it to build up big decoder larger decoder.

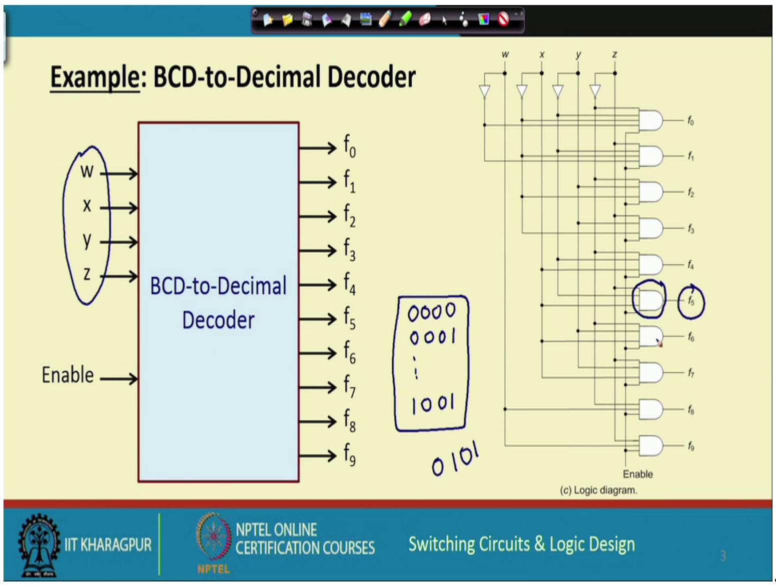
Like the example we showed that using 2 to 4 decoder we can construct 4 to 16 decoder. Now, here we shown an alternate approach here also we are trying to build a 4 to 16 decoder, but the approach is different. So, here I am showing this schematic so, we are using two 2 to 4 decoders. If the concept is that I mean using a row column concept, there are 4 columns and there are 4 rows.

One of the decoder depending on two of its inputs will be selecting one of the rows. The other decoder depending on the other two inputs will be selecting one of the columns. Suppose you applied W X equal to 1 1 which means the 4th row is selected and, if we apply W Z equal to 1 0 which means third row is selected.

So, it is 1 1 and 1 0 so, this last row and third column means I am referring to this junction, now what I do at every junction I connect an AND gate you see, every row line and every column line I connect as inputs to an AND gate and the output of the AND gate are my decoder outputs. So, just imagine here 1 1 and 1 0 will what if it is 1 1, then only the last row will be 1, if it is 1 0 the third column will be 1.

Now, if I connect through these two lines and AND gate, both of this input will be 1 and 1 I have connected this and this. So, the output will also be 1, but for all other AND gates this both inputs will never be 1 at least one of them will be 0. So, all others will be 0 only this will be 1. So, this works perfectly as a decoder depending on what I am applying to the input W X and Y Z exactly one of the gate outputs is becoming 1, that is my decoder output normally large decoders are built, or constructed using this way, because here as you can see the number of gate inputs are all small all two input and gates you required right ok.

(Refer Slide Time: 04:45)



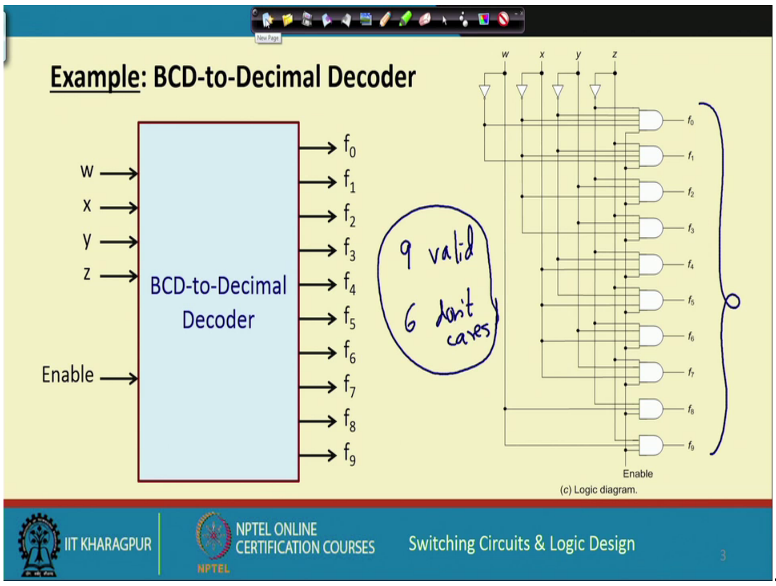
So, next move on so, let us talk about another kind of a special decoder, which is sometimes very useful particularly, when we work with BCD numbers, we need to use a decoder to identify what decimal digit, or BCD digit is that we need a decoder which is called a BCD to decimal decoder.

BCD as you know, BCD is a 4 bit number encoding, that is a W X Y Z is BCD and in BCD number the valid numbers go from 0 0 0 up to 9, the remaining 6 combinations are considered to be not valid. So, in the output we have only 10 outputs not 16 outputs like in a normal decoder, depending on whether you are applying one of these so, one of the outputs will be one so, again there is an enable.

So, the way you construct the decoder is exactly same as the normal decoder, let us say this 5, the gate for this 5 this 5 indicate 5 is 0 1 0 1. So, we will be connecting using first one to connect enable, then you connect X, then you connect Y bar, then you connect Z and of course, W bar.

So, this way you can connect this inputs in various way and of course, here what final realization we have shown is after minimization, because you see in this case among the inputs there are 9 valid inputs and they were 6 don't cares.

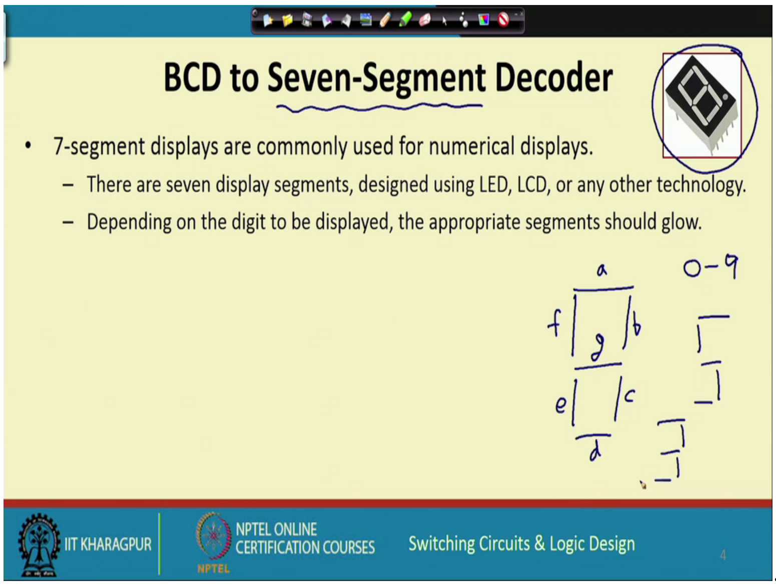
(Refer Slide Time: 06:47)



Example: BCD-to-Decimal Decoder

So, when you do a minimization these don’t care inputs also will result in the cancellation of the some of the terms. So, I am showing just the final realization, I am not showing in the steps of minimization. So, this is another kind of a decoder that we use in some designs BCD to decimal decoder.

(Refer Slide Time: 07:17)



BCD to Seven-Segment Decoder

- 7-segment displays are commonly used for numerical displays.
  - There are seven display segments, designed using LED, LCD, or any other technology.
  - Depending on the digit to be displayed, the appropriate segments should glow.

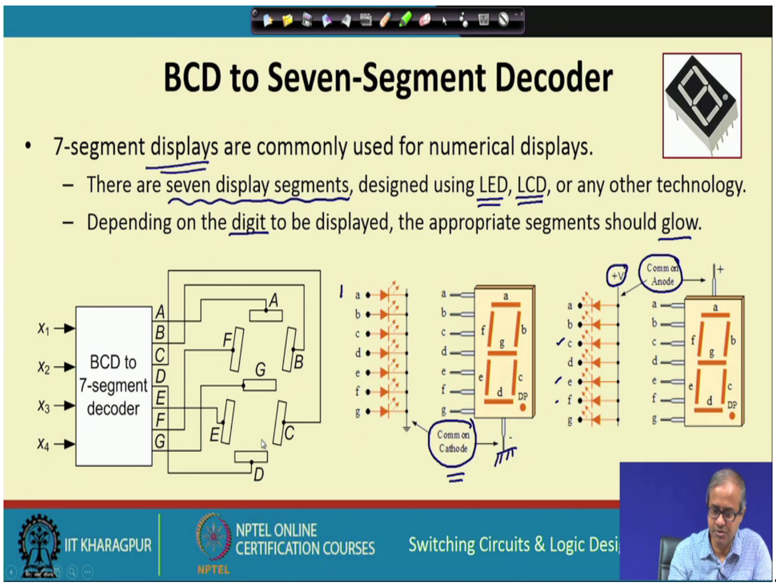
a
b
c
d
e
f
g
0-9
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

There is a another kind of a decoder which is also very useful well we have also, we have all seen this kind of digital displays in many gadgets and places in a in our calculators, in our washing machines, in our refrigerators different places, where there are some digitals displays the displays that display the number of displayed in a very specific way these are called 7 segment displays.

So, how 7 segment display you see here I have shown a picture of a typical of 7 segment display unit, you see 7 segment display unit there are 7 lamps, which are typically numbered or named as a b c d e f g by selectively drawing this lamps, you can display any of the digits from 0 to 9. Let us see if you want to display 5 you display this digits this lamp this is 5. So, if you want to display 3 you display these 0 to 9, you can display anything right.

So, basically this 7 segment display units that we talk about these displays, there are 7 display segments that that I have shown here and, the way they are manufactured that can be different, they can be manufactured using light emitting diode, liquids crystal display, neon lamp or any other technology. So, as I have said the examples depending on what digit you want to display, the appropriate segment should be activated and they should glow ok. So, these displays come into variety.

(Refer Slide Time: 09:16)


BCD to Seven-Segment Decoder

- 7-segment displays are commonly used for numerical displays.
  - There are seven display segments, designed using LED, LCD, or any other technology.
  - Depending on the digit to be displayed, the appropriate segments should glow.

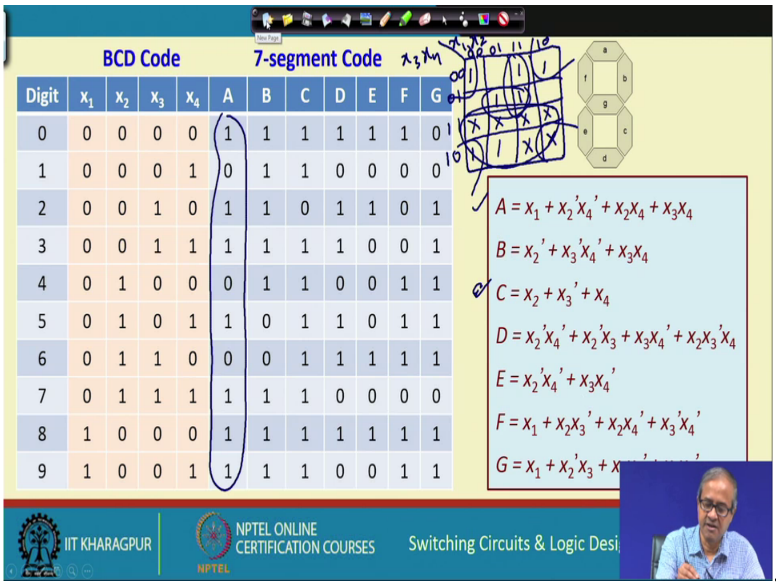
Let us talk about light emitting diode LED kind of displays, the first variety is the common cathode variety, where this 7 segment are actually 7 LEDs, or light emitting diodes, they are called common cathode as the cathode terminals of the 7 LEDs are tied together, this is the common terminal which you connect to ground.

And you apply a voltage to the this a b c d e f g input depending on which one you want to glow ok. Similarly you can have a common anode kind of a display unit, where the LEDs are connected in reverse order, the anode point is connected together and, they are connected to a positive supply voltage and, when you want to glow a digit you will have to ground the corresponding input so, that a current flows ok.

Now, in the example that you will be showing will be assuming that is common cathode that means, if you want to display a segment you have to apply a high voltage let us say 1 on that line, the other side is grounded. So, when you talk about BCD to 7 segment decoder we are talking about a circuit like this.

So, we want to design a decoder this, where there will be 4 input applied this is our BCD input and 7 segment outputs will be generated which will be connected to the 7 segments of this display A B C D E F G and depending on the BCD input the correct segments will be activated so that the corresponding digit blows this is a basic idea.

(Refer Slide Time: 11:25)



	BCD Code				7-segment Code						
Digit	x_1	x_2	x_3	x_4	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

$A = x_1 + x_2'x_4' + x_2x_4 + x_3x_4$

$B = x_2' + x_3'x_4' + x_3x_4$

$C = x_2 + x_3' + x_4$

$D = x_2'x_4' + x_2'x_3 + x_3x_4' + x_2x_3'x_4$

$E = x_2'x_4' + x_3x_4'$

$F = x_1 + x_2x_3' + x_2x_4' + x_3'x_4'$

$G = x_1 + x_2'x_3 + x$

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

So, the truth table the BCD to 7 segment decoder looks like this. So, I have also shown a 7 segment display here side by side for reference, let us say my digit is 0. So, when I want to display 0 other than g all the other segment should be 0. So, you see a b c d e f are all 1 only g is 0. So, if I want to display 1 only B and C must glow all other should be 0, you see only B and C are 1, for 2 a b g e n and c should glow A F and C should be off you see for 2 C is off and F is off.

So, like this you can display all the digits say for 8 all the segments should glow all are 1 for 9 only E and D should be off, for 9 only E and D should be off other should glow. So, it is easy to construct this table right corresponding to the BCD input code. So, exactly how I you want to display it so, you can just activate the segment assuming common cathode display that 1 means it will glow 0 means it will not glow ok.

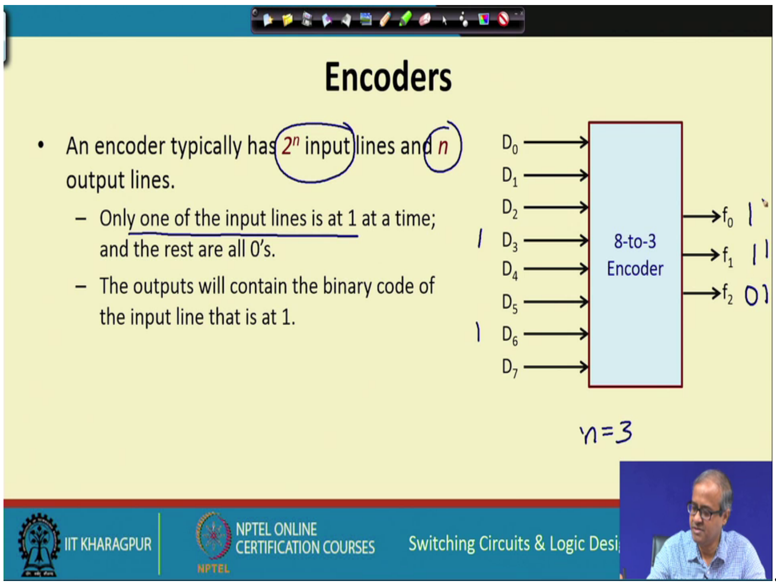
And, just from this you can write down the expressions also, I have not shown you the exact process of minimization again well I leave, this again exercise for you this a 4 variable function there are so, many don't care states 10 11 12 13 14 15, there are 6 don't care states. So, when you if you I mean when you use a Karnaugh map you can use this don't care states also for minimization, let me take one example as an instant sorry yeah, let us take one example let us consider a 4 variable Karnaugh map, let me show you one x 1 x 2 x 3 x 4 let us say x 1 x 2 is in this direction, x 1 x 2 and x 3 x 4 in this direction let us say.

So, the inputs as 0 0 0 1 1 1 1 0 and 0 0 0 1 1 1 1 0, let us take this segment A for example, segment has many Karnaugh let us take another let us say segment C, C can be minimized let us have C look at the segment C, or fine fine A is fine let us take a look at the segment A.

You see where they are 1 for 0 this is 0 1 2, 2 is 0 1 2 is this, then 3, 3 is also 1 this is 3 4 is 0 5, this is 4 this is 5 5 6 7 4 4 5 6 7, this is 7 and 8 and 9 this is 8 and this is 9, this is your Karnaugh map and in addition there are 6 don't cares 10, 11, 12, 13, 14, 15, 10 is 1, 0 so, I am showing the don't care as x these are all don't cares, these 6 are don't cares.

So, now, you will have to just include these once and go ahead to the minimization. So, you can make the cubes like here the 4 corners will be a cube, then you can have a cube like this, you can have a cube like this, you can have a cube like this, you can have a cube like this, like if this if you do a minimization we will be arriving at this expressions ok, I am not working all of them I am leave it as an exercise for you. So, you can actually verify this ok, minimize this and see that whether it is a expressions are tallying with this ok.

(Refer Slide Time: 16:16)



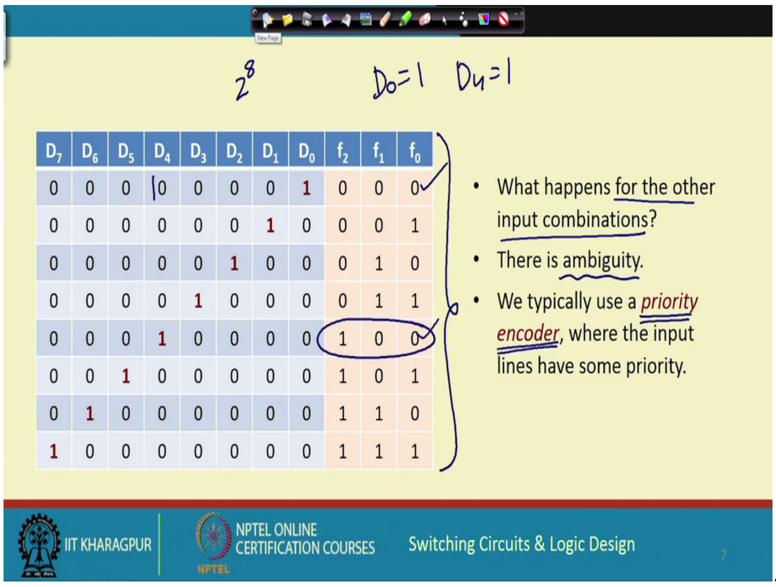
Encoders

- An encoder typically has 2^n input lines and n output lines.
  - Only one of the input lines is at 1 at a time; and the rest are all 0's.
  - The outputs will contain the binary code of the input line that is at 1.

Let us move on next let us come to a circuit, which you can call it is an reverse of the decoder, I call it an encoder, there are many inputs, there are few outputs, depending on which input is activated at a time I am assuming only one of the input is active.

So, the output will contain the binary code of the input which input that is active, let us see how it works. So, an encoder will typically have 2 to the power n input lines and n output lines. So, here there is an example for n equal to 3, there are 8 input lines and 3 output lines. So, I am assuming here that only one of the input lines is one at a time. Let us say D 3 is 1 the output will contain the binary code of D 3, 3 is 3 so, it is 0 1 1 that is 3 this will be the output, if this 6 is 1, 6 is 1 1 0 so, it will be 1 1 0 and so on this is how the encoder will work.

(Refer Slide Time: 17:41)



2^8
$D_0=1$ $D_4=1$
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	f_2	f_1	f_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

- What happens for the other input combinations?
- There is ambiguity.
- We typically use a *priority encoder*, where the input lines have some priority.

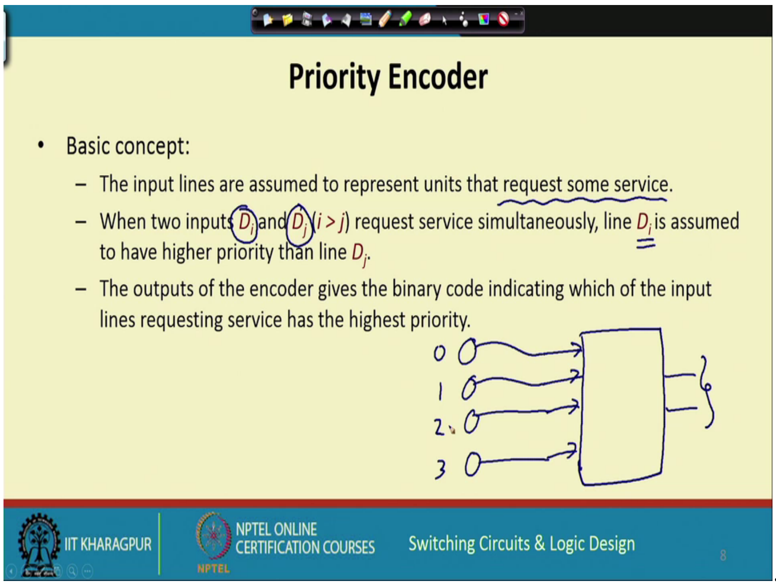
So, for a normal encoder the truth table will look like this. So, this is the first case I am show, if you see here again you need not show all the there are 8 inputs. So, there will be 2 to the power 8 possible input combinations. So, you need not show the whole truth table, because this is a functional block you can show only the functional rows. The first row says only this row is 1 others are all 0s. So, this will mean 0 0 0, if D 1 is 1 0 0 1, D 2 is 1 0 1 0 like this, where the other inputs are not valid and they are all don’t cares.

Now, now one issue here is that in this kind of an encoder design that, we are assuming that exactly one of the input is at 1, but let us say by accident or otherwise I have also made another input as 1. Let us say D 4 is 1 D 0 is 1. So, what will be my output? So, will it be 0 0 0, or will it be corresponding to D 4 1 0 0.

So, for the other input combinations where more than one inputs can become 1, there is some ambiguity. So, to remove this ambiguity normally the encoders that we design, they are something called priority encoders not a normal encoder like this table shows, now in a priority encoder the input lines will have some priority like. Suppose, I have said D 0, I have applied 1 D 4 also I have applied 1 and I am saying line that the D 4 is having higher priority than D 0.

So, if both of them are 1 D 4 will take priority and the output corresponding to D 4 will be generated 1 0 0 this is how the priority encoder should work. So, for a priority encoder the basic concept is like this let me just repeat.

(Refer Slide Time: 20:00)



Priority Encoder

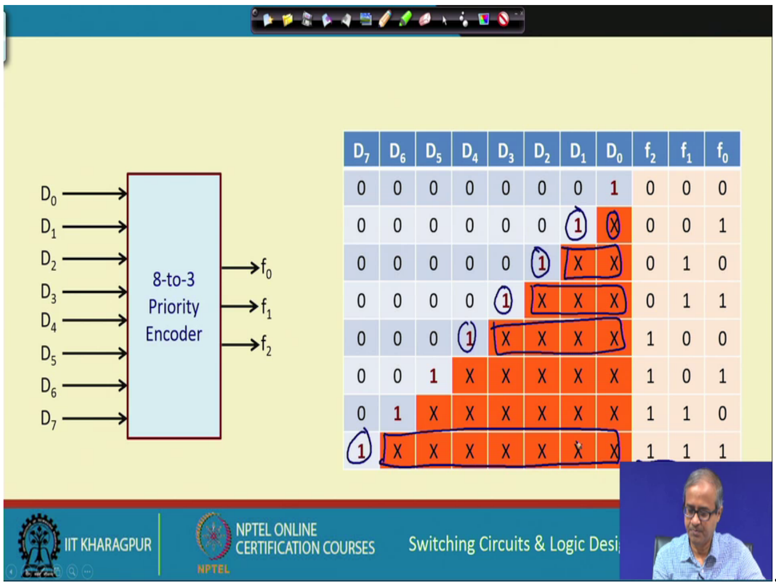
- Basic concept:
  - The input lines are assumed to represent units that request some service.
  - When two inputs D_i and D_j ($i > j$) request service simultaneously, line D_i is assumed to have higher priority than line D_j.
  - The outputs of the encoder gives the binary code indicating which of the input lines requesting service has the highest priority.

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

Now, let us put it in the proper context, see a priority encoder, or means any encoder circuit, they are used an application. Let us say 4 inputs and 2 outputs, they are used an application where some sub circuits, or some devices or some units they are generating some kind of requests, the input lines are assumed to represent you need that requests some service.

And the output will indicate that which one of the inputs that the input 0 1 2 3 there are 4 input units, which of the inputs are generated the request. Now, in a priority encoder as I said whenever 2 of the inputs let D i and D j, where i is greater than j request service simultaneously, then you assume that D i is having higher priority, because i greater than j. So, the output will be generating the binary code for the highest priority input that is active. So, this is how the priority encoder works ok.

(Refer Slide Time: 21:22)

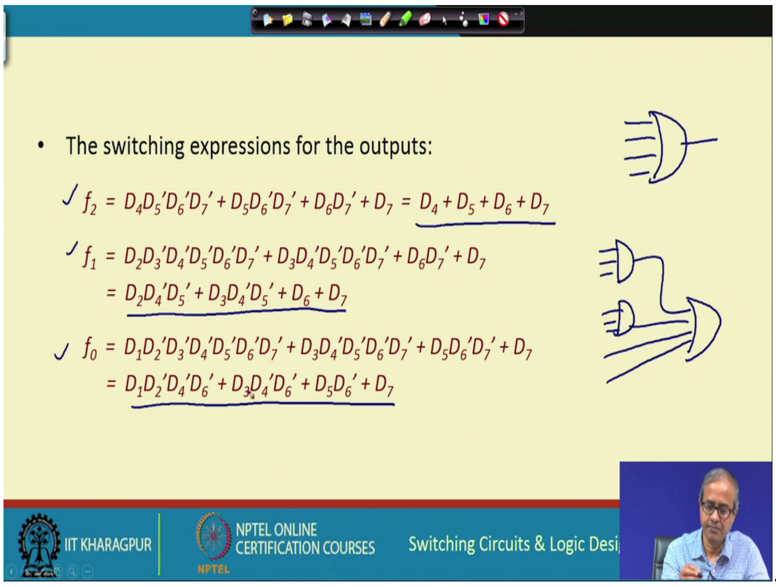


D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	f_2	f_1	f_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

Let us look at the truth table it will be clear think of an 8 by 3 priority encoder there are 8 inputs, there are 3 outputs. For the first case where only D 0 is 1 others are all 0s, there is no ambiguity it is 0 0 0, but when D 1 is 1, D 1 has a higher priority than D 0. So, D 0 will be don’t care D 0 can be 0 or 1 does not matter, but D 1, but the rest are 0s it will be 0 0 1, but when D 2 is 1 the lower priority 1s will be don’t care, it will only be important the that the higher priority 1s are 0 0 0.

Similarly, D 3 is 1 lower priorities are all don’t care this is like this so, the truth table you look like there will be lot of don’t care entries here. So, when D 7 is one for example, so, the other inputs does not influence the output. So, if D 7 is 1 and the output will always be 1 1 1 irrespective of the other inputs, right.

(Refer Slide Time: 22:47)



- The switching expressions for the outputs:

✓ $f_2 = D_4D_5'D_6'D_7' + D_5D_6'D_7' + D_6D_7' + D_7 = \underline{D_4 + D_5 + D_6 + D_7}$

✓ $f_1 = D_2D_3'D_4'D_5'D_6'D_7' + D_3D_4'D_5'D_6'D_7' + D_6D_7' + D_7$

$= \underline{D_2D_4'D_5' + D_3D_4'D_5' + D_6 + D_7}$

✓ $f_0 = D_1D_2'D_3'D_4'D_5'D_6'D_7' + D_3D_4'D_5'D_6'D_7' + D_5D_6'D_7' + D_7$

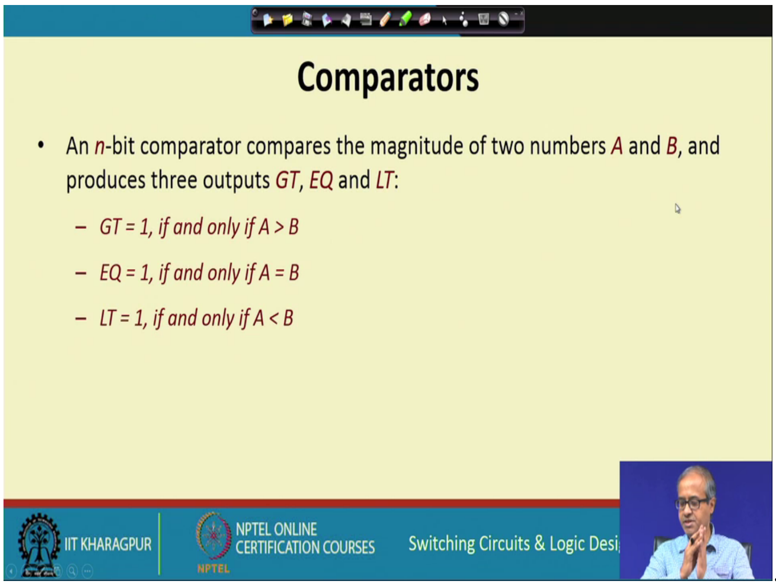
$= \underline{D_1D_2'D_4'D_6' + D_3D_4'D_6' + D_5D_6' + D_7}$

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

This is how priority encoder works then it is again I leave it as an exercise for you to find out how to generate the expressions, just here I am showing out the expression for the outputs. So, for this f 2 f 1 and f 0 if you write down the expressions, then after minimization the expressions will be like this.

So, I leave it as an exercise for you to verify again. So, if you want to implement this using gates will of course, first you will have to minimize this and then implement these using gates right ok. So, f 2 for example, you require a single OR gate nothing else is required f 2 will required a single 4 input OR gate 4 5 6 7, this f 1 for example, will be requiring 2 AND gates 3 inputs each, well for see a some not gates also some are bar are the D 4 D 5 bar and D 6 D 7 are direct. So, there will an OR gate correct D 6, D 7 will be direct hm.

(Refer Slide Time: 24:00)



Comparators

- An *n*-bit comparator compares the magnitude of two numbers *A* and *B*, and produces three outputs *GT*, *EQ* and *LT*:
  - *GT = 1, if and only if A > B*
  - *EQ = 1, if and only if A = B*
  - *LT = 1, if and only if A < B*

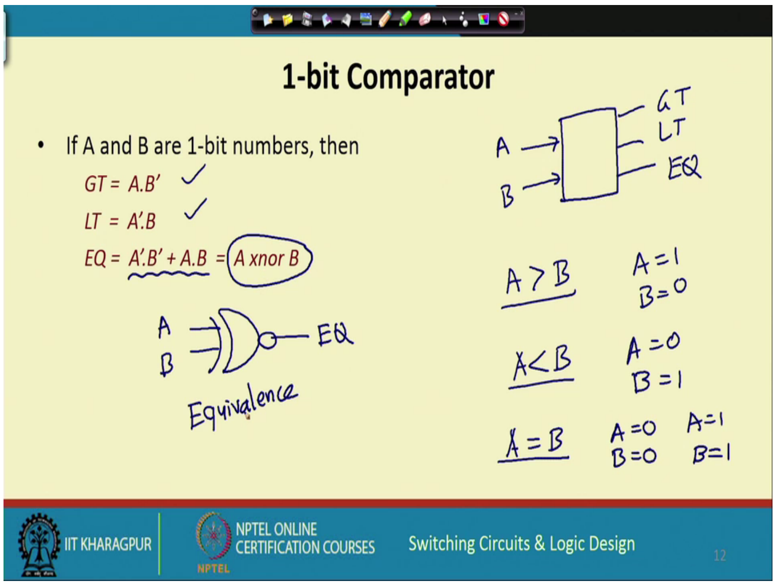
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Switching Circuits & Logic Desi

.

So, similarly f 0 now, let us talk about a circuit element called comparator which is also very important, sometimes we need to compare the values of the magnitudes of two numbers, whether they are equal whether one is greater than the other, or whether one is less than the other, this is called a magnitude comparator a simple comparator digital comparator.

So, when you talk about an n bit comparator, it basically compares the magnitude of two n bit numbers A and B and, there will be 3 outputs GT EQ and less than greater than or equal to less than, where GT will be said to 1, if and only if A is greater than B in magnitude EQ will be 1 LT will be 1, if A is less than B, this is how the comparator works right.

(Refer Slide Time: 25:00)



1-bit Comparator

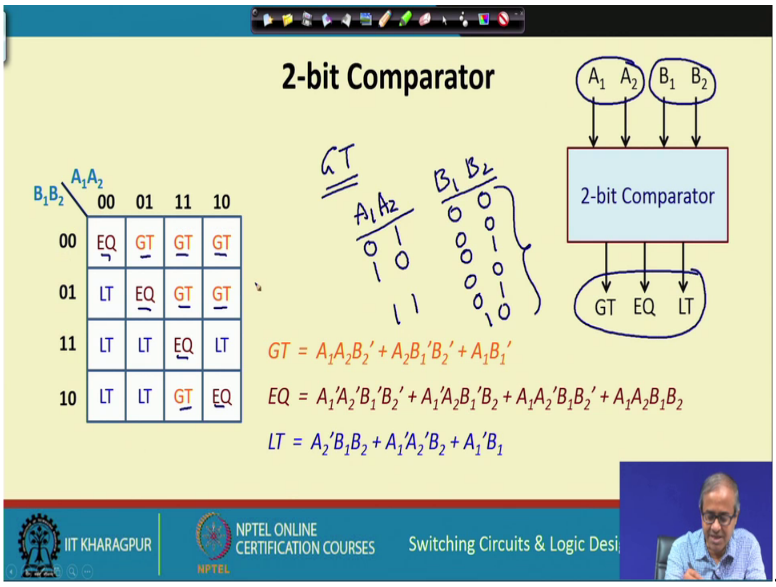
- If A and B are 1-bit numbers, then
  - $GT = A.B'$ ✓
  - $LT = A'.B$ ✓
  - $EQ = \underbrace{A'.B' + A.B} = \boxed{A \text{ xnor } B}$

So, let us it is the very simple scenario first, just a one bit comparator. So, I want to design a circuit, where the two numbers are adjust single bit numbers A and B 1 bit number, and I have my three outputs greater than less than and equal to.

Now obviously, a greater than B means what for a single bit number that A is 1 and B is 0 this is only case. So, it will be A B bar A B bar less than B means so, here also only case only one case A 0 B 1 so, A bar B A bar B and equal case A equal to B you see that can be two scenarios, A 0 B 0, or A 1 B 1.

This can be expressed as A bar B bar or A B which is nothing, but the exclusive nor function, symbolically it is written like this exclusive nor, if A B are the inputs output will be this EQ. Because of this property x nor is basically checking equality of 2 bits, this is sometimes also known as the equivalence function, or the equivalence gate x nor is sometimes also known as equivalence gate, 1 bit comparator is very easy to design, let us look into larger size comparators, let us say 2 bit comparator.

(Refer Slide Time: 26:54)



2-bit Comparator

B_1B_2 \ A_1A_2	00	01	11	10
00	EQ	GT	GT	GT
01	LT	EQ	GT	GT
11	LT	LT	EQ	LT
10	LT	LT	GT	EQ

$GT = A_1A_2B_2' + A_2B_1'B_2' + A_1B_1'$

$EQ = A_1'A_2'B_1'B_2' + A_1'A_2B_1'B_2 + A_1A_2'B_1B_2' + A_1A_2B_1B_2$

$LT = A_2'B_1B_2 + A_1'A_2'B_2 + A_1'B_1$

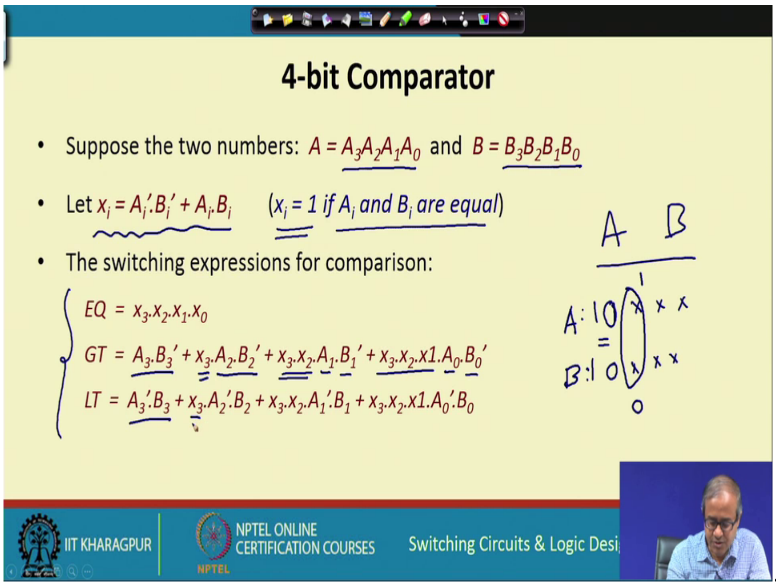
Here I am assuming that the numbers are 2 bits A 1 A 2 is 1 number B 1 B 2 is another number and I want to generate the 3 outputs, you see let us construct the Karnaugh map this A 1 A 2 along this side B 1 B 2 along this. Let us think of the 3 scenarios separately greater than so, what are the greater than scenarios.

Let us take some examples A 1 A 2 and B 1 B 2, it is a when A 1 A 2 is let us say 0 1 and B 1 B 2 is 0 0, then A is greater than B so, when it is 1 0 B 1 B 2 is either 0 0 or 0 1, then also A is greater than B and A 1 A is 1 1 B 1 B 2 is either 0 0, or 0 1 or 1 0 then also it is greater so, there are 6 possible scenarios.

So, see in the truth table I am showing this a 6 possible cells, where GT GT GT or not, because GT LT and EQ will be all exclusive that is why I can show it on the same truth table, equal to will be 0 0 equal to 0 0 0 1 equal to 0 1 1 1 equal to 1 1 1 0 equal to 1 0 and the remaining cells will be all be less than.

So, directly from the single truth table you can actually write down the expression for all of them, like for the GT the cubes will be one will be this one will be this and the other will be this these two and so, on. So, if you write down it will be this. This EQ it cannot be minimized any further, there will be 4 terms with all containing 4 letters, for LT also you can combine this 4, you can combine this 2, you can combine this and this and there will be three terms right. So, a 2 bit comparator is also relatively easy to design you can do it in this way.

(Refer Slide Time: 29:19)



4-bit Comparator

- Suppose the two numbers: $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$
- Let $x_i = A_i'.B_i' + A_i.B_i$ ($x_i = 1$ *if* A_i *and* B_i *are equal*)
- The switching expressions for comparison:

$$EQ = x_3.x_2.x_1.x_0$$

$$GT = A_3.B_3' + x_3.A_2.B_2' + x_3.x_2.A_1.B_1' + x_3.x_2.x1.A_0.B_0'$$

$$LT = A_3'.B_3 + x_3.A_2'.B_2 + x_3.x_2.A_1'.B_1 + x_3.x_2.x1.A_0'.B_0$$

A B
A: 1 0 x x x
B: 1 0 x x x
1
0
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Now, when we talk about a 4 bit comparator let us see, suppose I want to design a 4 bit comparator, there are two 4 bit numbers B 0 to B 3 A 0 to 3 let us say. Let us denote or introduce a symbol called x i which is the equivalence of A i and B i that means, xnor function that means, x i will be 1 if A i and B i are equal.

Now, with this notation we can directly write down the expressions for EQ GT and LT, you see A and B they will be equal if all corresponding bits are equal; that means, x 3 is 1, x 2 is 1, x 1 is 1 and also x 0 is 1 just end up all 4, G 3 you see you think of the two numbers the number A and B.

So, if you see that A is 1 that A 3 is 1 and B 3 is 0, then you did not have to see the other bits, because irrespective of the other three bits will be the most significant bit A is greater than B, then always A will be greater A 3 B 3 bar, but if you find they are equal these two are equal 0 0 or 1 1 these two are equal, then you need to see the next bit.

If this is 1 and this is 0 so, you see the first bit is equal and A 2 is 1 B 2 is 0, or the first 2 bits are equal A 1 is 1 B 1 is 0, or the first 3 bits are equal A 1 is 0 B 1 A 0 is 1, B 0 is 0 the LT is just the reverse instead of A 3 B 3 bar is A 3 bar B 3 bar, this is 0 1, they are equal and this they are equal, then this they are equal then this same way you can straight away write down.

So, you see for this kind of functional blocks you did not always have to construct the truth table to write down the expression, you think mentally that what this block is supposed to do and you can functionally you can write down the switching expression directly. This often we do for the many of the functional books. So, with this we come to the end of this lecture on logic design, in our next lectures we shall be looking at various ways of representing Boolean functions and some of the ways in which you can manipulate them.

Thank you.