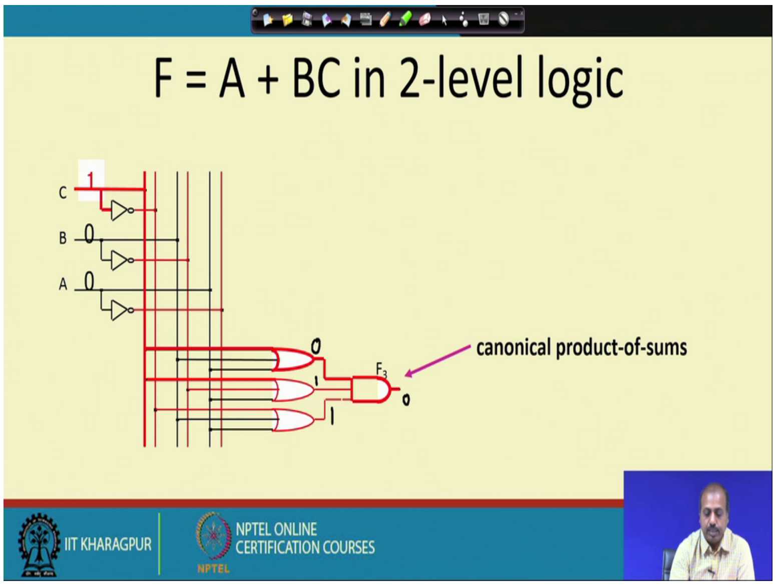
Digital Circuits
Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 19
Logic Gates (Contd.)

So consider the function F equal to A plus BC when it is realized in canonical product of sum form; and this is the realization, ok.

(Refer Slide Time: 00:19)



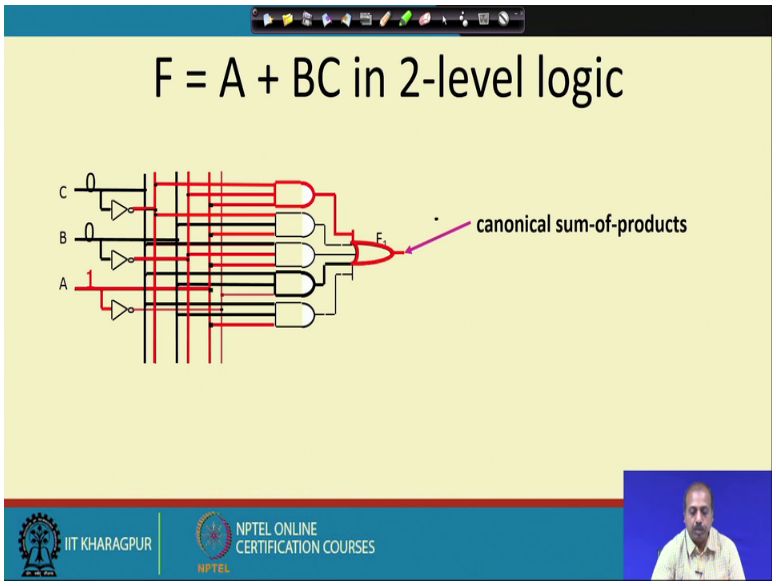
F = A + BC in 2-level logic

Now suppose this ABC all of them are at 0. So, what are what has happened is that all of them being at 0. So, here this OR gate, all the 3 inputs are all the 3 inputs are 0 so, this output is 0. And this OR gate, this has got sum of the inputs has 1, like when B is 0 so, this input is 1. So, this is 1 and similarly, this is also 1, because this input is coming here as a 1 so, this AND gate output is equal to 0.

Now, now consider the case when this A this C value it has changed from 0 to 1. When it has changed from 0 to 1 so, we have, when this C value has changed from 0 to 1 so, this output is I have this C, C input has become equal to 1, as a result this OR gate will now be equal to 1. So, this output becomes equal to 1. And after sometime this inverter output will be activated. So, inverter has got some delay. So, before the inverter delay was activated so, this OR gate was remaining at 1 so, you are getting a some output.

But now after sometime this inverter will be equal to 0, as a result this OR gate value will be become equal to 0 so, this x 3 will be will be equal to 0, ok. So, this way due to the presence of these delays of the inverters, ok, so, this circuit will produce a glitch at the F 3 output, ok.

(Refer Slide Time: 02:20)



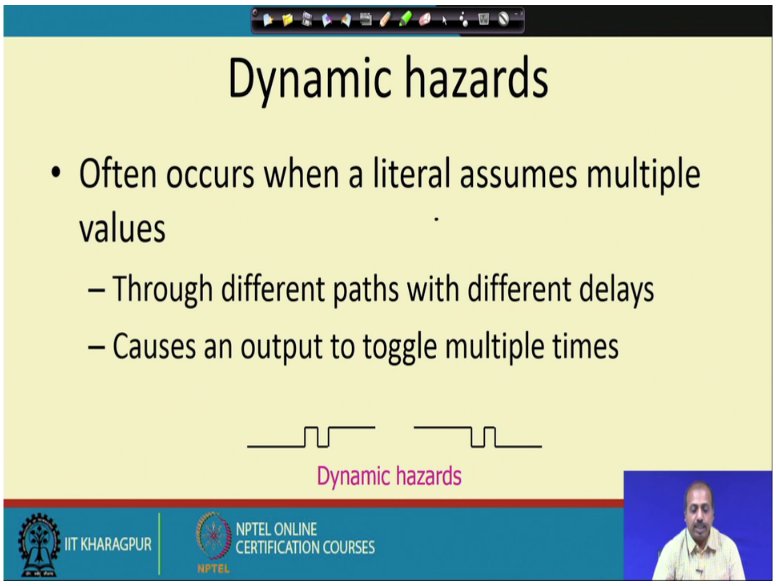
F = A + BC in 2-level logic

So, this, similarly this F 1. So, F 1 suppose this ABC the A is 0 B and C are 1. So, this is the situation the F 1 output is equal to 1.

And now if there is a change that B and C become equal to 0 and A changes to 1. Then what will happen is that due to the delay, so, the effect will not come immediately. So, F 1 will be remaining as equal to 1, and now it will become equal to 0. And then after sometime B suppose this inverter of B, it changes it is state. So, it is get you 0 so, it will be getting as the it will affect the circuit after that. So, this way so, if you look into the gate delays into if you take this gate delays into consideration, you may find some behavior of this circuits in some intermediary form, so, that is not a that is not stable actually so, in between it shows some glitch, so, that can happen.

So, it is just you can just trace through this circuit to see that this is changing the value, ok. So, whenever it is red means it is going to logic 1, and wherever I am showing it by black line means it is at logic 0, ok. So, you follow that convention and trace through this diagrams, then you will find the implementation this creation of the glitches.

(Refer Slide Time: 03:37)


Dynamic hazards

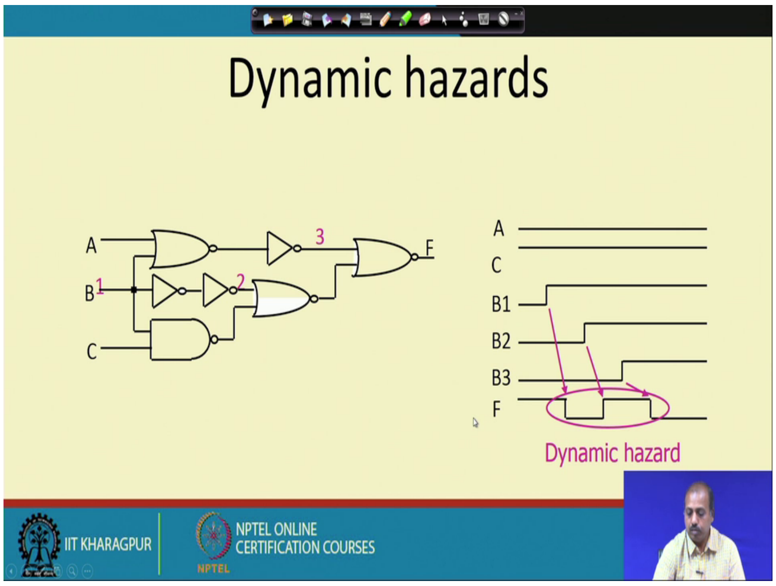
- Often occurs when a literal assumes multiple values
  - Through different paths with different delays
  - Causes an output to toggle multiple times

Dynamic hazards
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

On the other hand this dynamic hazard so, they occur when a literal assumes multiple values, ok. So, through a different through different paths, different delays, they have got different values, and causes an output to toggle.

So, as we have seen the shown the output should go from 0 to 1, but in between it make some toggle, or it goes from 1 to 0 in between it makes some toggle.

(Refer Slide Time: 03:59)


Dynamic hazards

So, consider this circuit that has got this ABC as a input, and this suppose this B value. So, the so, suppose situation is like this, that A is at logic low C is at logic high. And this

B B point at A B 1 at A B 1 it changes a state from a 0 to 1; so, that B 1 it changes the state.

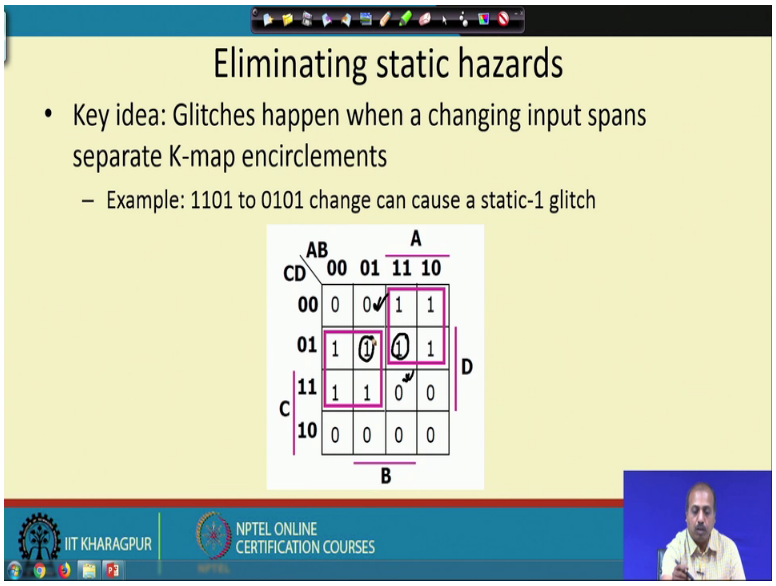
Then what will happen? So B 2 so, at this point so, there are 2 inverters from B 1 to B 2 so, after 2 inverter delays so, B 2 will show this waveform, at this point. Then at B 3 so, B 3 has got so, B so, A is at logic low so, for this nor NOR gate so, it is nothing but an inverter so, this B 3 is at so, B 1 sees the delay of this NOR gate plus this inverter to produce B 3. So, that is a, if I assume that this NOR gate has got delay equal to 2 inverters. So, after 3 delays so, you or after so, they are they are having some same delay. So, NOR gate and this inverter they are having same delay.

So, after 2 gate delays so, B 3 gets the value of B 3 gets the value of B 1, so, B so, B 1 has changed from 0 to 1. And B 3 has changed from 0 to 1 after 2 gate delays. Now when it comes to this F and this C is always 1 ok. So, C is always one means this um this NAND gate it is it is always giving B 1 bar, and then this is giving me the whatever is coming here depending upon this B 2 value so, this output of the NOR gate is coming.

Now, if you trace this circuit, again you can find that the situation, will be like this that initially the F will be high, it will show be low for one gate delay, then it will be high for 2 gate delays, and then again it will be low for the remaining part of the system. So, that way so, there is a glitch so, this is the dynamic hazard here. So, the it shows a dynamic hazard at this point, so, this there is a dynamic hazard at this point. So, we can just trace through this circuit. And see that this type of timing diagram this waveforms are appearing, and that creates the dynamic hazard.

So, this has to be avoided, now how do we do that? So, this is the dynamic hazard.

(Refer Slide Time: 06:25)



Eliminating static hazards

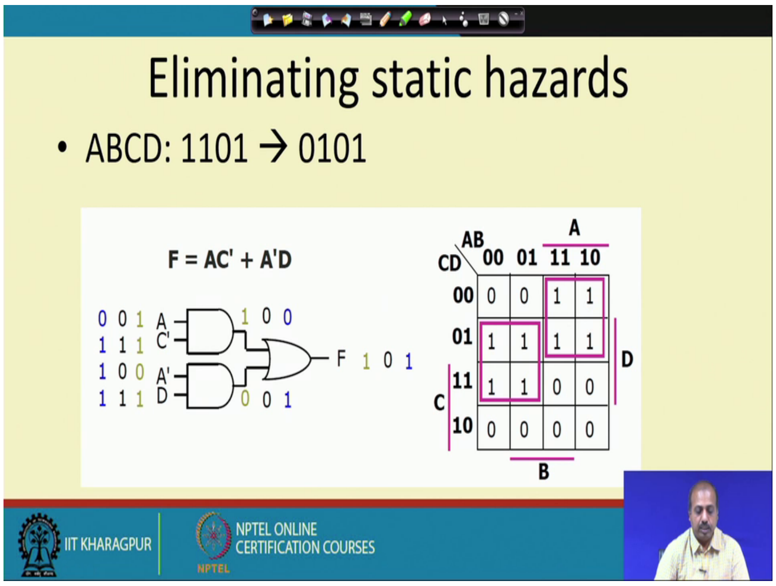
- Key idea: Glitches happen when a changing input spans separate K-map encirclements
  - Example: 1101 to 0101 change can cause a static-1 glitch

CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

Now, how can we; actually this dynamic hazards are difficult to remove, ok. So, so we will be talking about static hazard first. So, the key idea for eliminating a static hazard is that, the glitches happen when a changing input spans separate K map encirclements. So, if some so, if you are going from say this box to this box, ok. So, we are changing some inputs so, that it goes from this circle to this.

So, in between it may go through some 0 state, as a result a circuit output may become 0 so, that is the key point, ok. For example, so, when you are going from 1 1 0 1 to 0 1 0 1. So, 1 1 0 1 is this one 1 1 0 1 is this one to 0 1 0 1 so, 0 1 0 1 is this one, ok. So, whenever you are going from here to here. So, it may so happen that this delay comes so, A is delayed a bit and then this a 1 to 0 so, that delay takes some time, and in between it goes to this 0 state or this 0 state and that way it creates a it creates a glitch, ok.

(Refer Slide Time: 07:48)



Eliminating static hazards

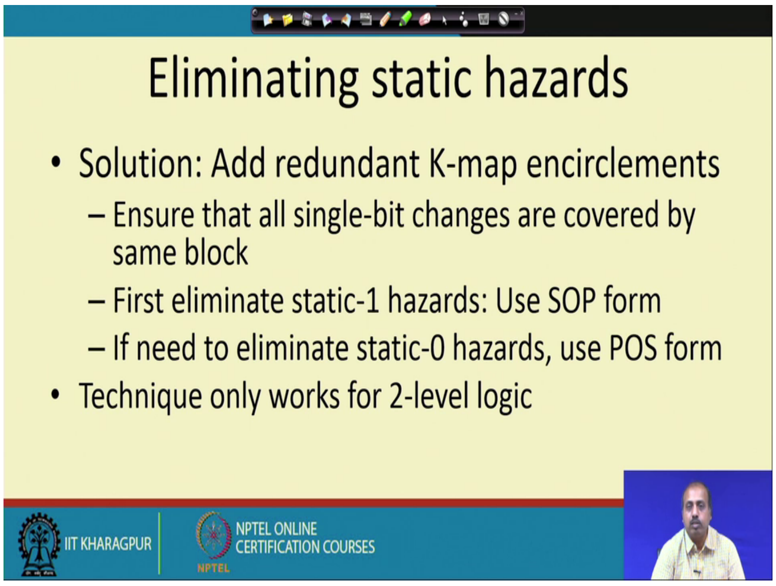
- ABCD: 1101 → 0101

CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

So, how to solve this? Sorry, so, how to solve this is something like this, that so, this is the situation like, say we have got the function so, this square it represents AC bar, and this other quad, it represents C A bar D, so, it is AC bar plus A bar D. Now if I have got the situation like ABCD is 1 1 0 1 so, you get this 1 1 0 1. So, A AB A A and C bar so, A and C bar are 1 1 and A bar and D. So, that is 0 1 so, it is applied here.

So, you get 1 and 0 here accordingly the output is 1. Now it is changing to 0 1 0 1, but this A changes, and it takes some time for A bar to change, as a result you may find that it will produce a 0 in between.

(Refer Slide Time: 08:35)



Eliminating static hazards

- Solution: Add redundant K-map encirclements
  - Ensure that all single-bit changes are covered by same block
  - First eliminate static-1 hazards: Use SOP form
  - If need to eliminate static-0 hazards, use POS form
- Technique only works for 2-level logic

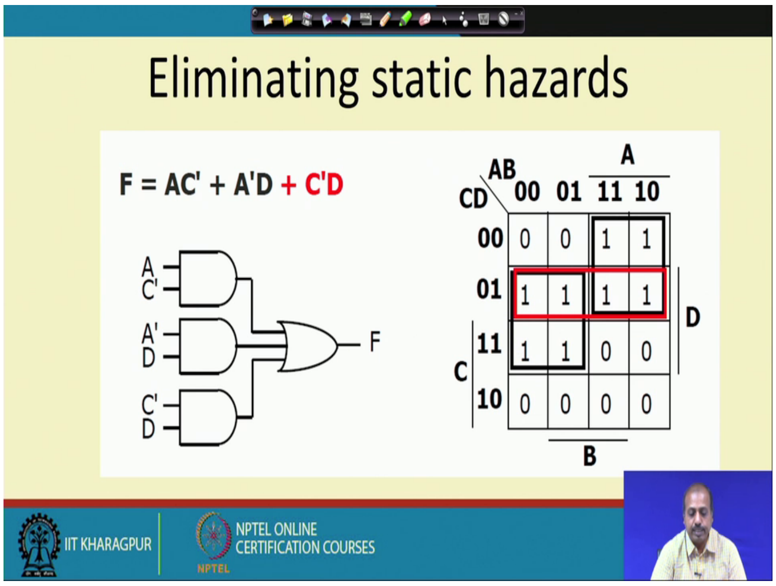
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, that 0 has to be avoided, how tosolve this is that, we add some redundant K map encirclements ok. So, we add some extra encirclements. So, like this so, we so, that this single bit changes can be covered in the the same block

So, it will eliminate static 1 hazard o, using sum of product form, and this if there is static 0 hazard is also there, then we have to use the POS form product of sum form also. So, use this so, first of all so, the process becomes complex, first of all you do encirclement and make the SOP form, and then if you find that this static 0 hazard is also there. Then you have to use this product of sum form and again do the encirclement in that fashion.

But unfortunately this technique works only for 2 level logic. So, multilevel logic there is no solution to this static hazard problem. So, this is the so, that if it occurs so, we have to be careful there.

(Refer Slide Time: 09:35)



Eliminating static hazards

$F = AC' + A'D + C'D$

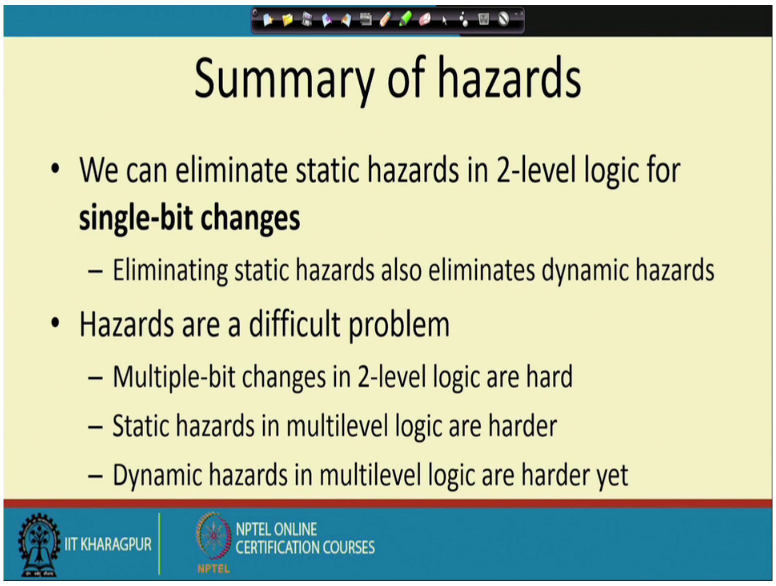
CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

So, this is an example so, how do we do this. So, the same circuit that A C bar plus A bar D, ok. So, what we have done, we have done some redundant encirclement ok. So, this C so, this was this red one, this is this red quad was not necessary, ok, because we have the all the ones are already covered by the remaining 2 quads.

But we do not, but we take this redundant one, what will happen is that even if this A is changing the value. Ok this C bar D so, this will hold the output ok; so, this is the problem occurred because for some amount of time this A and A bar both of them were equal to 0. As a result this OR gate input though the both the inputs were 0. But this C bar D so, C bar D term has been added. So, so C bar D does not make a change so, C bar D holds the value at 1. So, it will be we will be getting a one if you are changing the input combination like that, ok. So, this way we can eliminate the static hazard by doing some redundancy.

So, we are going a bit counter intuitive like for Boolean function minimization. So, we do not allow this type of extra extra quad or square formations. But in case of for eliminating this static hazard so, we are doing the other way. So, we are allowing this extra terms so that this hazards are not there.

(Refer Slide Time: 11:01)



Summary of hazards

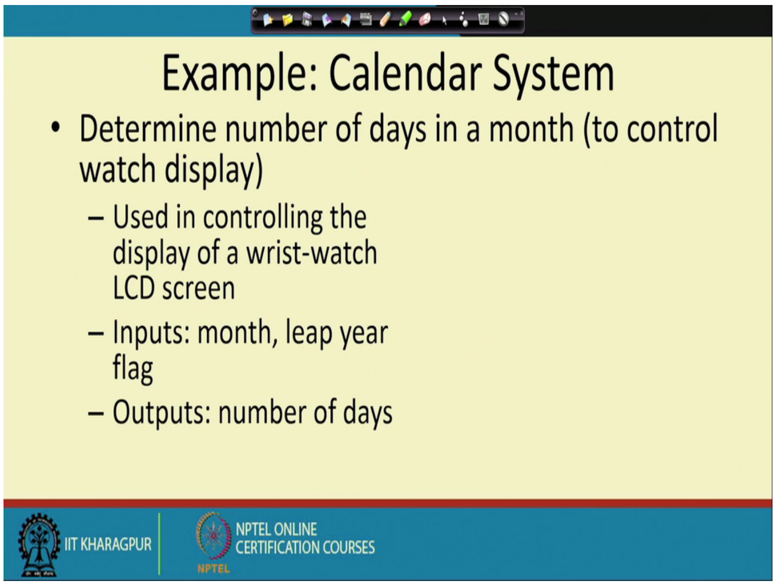
- We can eliminate static hazards in 2-level logic for **single-bit changes**
  - Eliminating static hazards also eliminates dynamic hazards
- Hazards are a difficult problem
  - Multiple-bit changes in 2-level logic are hard
  - Static hazards in multilevel logic are harder
  - Dynamic hazards in multilevel logic are harder yet

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NPTEL

So, we can eliminate static hazards in 2 level logic for single bit changes, and eliminating static hazards also eliminates dynamic hazards. So, that is one good thing that if we eliminate static hazards the dynamic hazards will also not be there; so, that way it is useful.

Hazards are a difficult problem so; multiple bit changes in 2 level logic are hard. So, we are a whatever example we are considering. So, we are considering only a single bit change, ok. So, if the multiple bits are changing so, it is difficult to solve the, that type of a hazards, and static hazard in multi-level logic is difficult, and dynamic hazards in multilevel logic are we can it can so, it is yet more harder, ok. So, that way it is difficult to solve this for multilevel circuit this dynamic hazard problem.

(Refer Slide Time: 11:52)



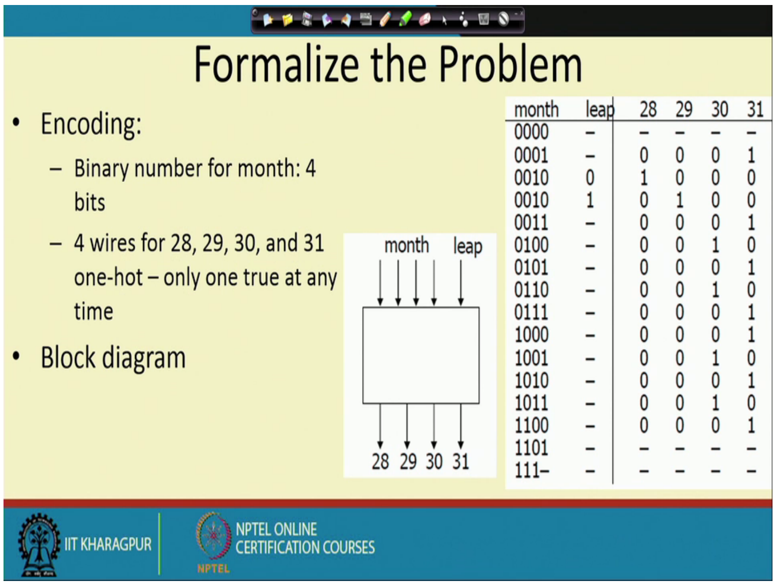
Example: Calendar System

- Determine number of days in a month (to control watch display)
  - Used in controlling the display of a wrist-watch LCD screen
  - Inputs: month, leap year flag
  - Outputs: number of days

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

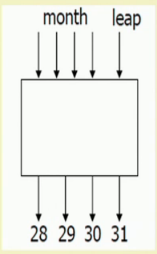
Next we will be looking into a few examples of this combinational logic design. So, suppose we have the first example we are determining number of days in a month to control some watch display. And it is used in controlling the display of a wrist watch LCD screen, inputs are month leap year a month and leap year flag, and outputs are the number of days. So, you give the month and the leap year flag it will output the number of days in it. So, you I have a combinational we I have to design a combinational circuit, a month will be one input and whether the year is a leap year or not. So, that will be coming as input and it will be outputting the number of days in the month.

(Refer Slide Time: 12:31)



Formalize the Problem

- Encoding:
  - Binary number for month: 4 bits
  - 4 wires for 28, 29, 30, and 31 one-hot – only one true at any time
- Block diagram



month	leap	28	29	30	31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

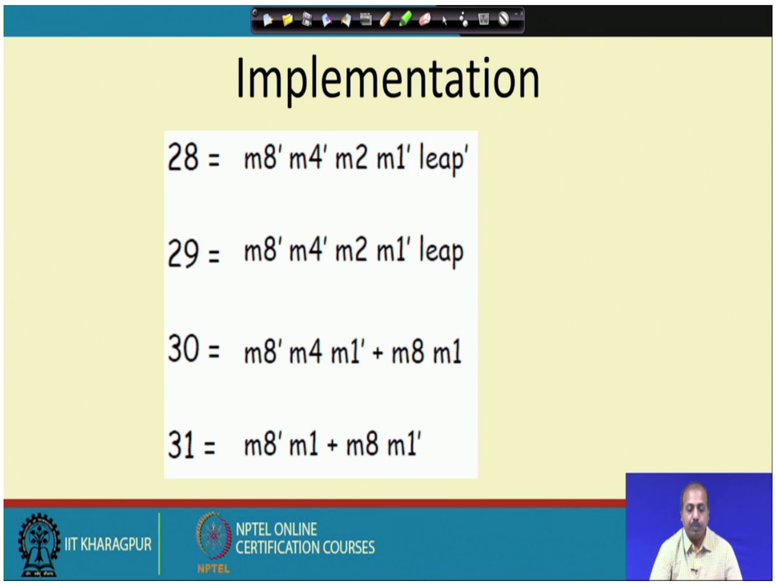
So, this is a typical example, see we have got this month can go from January to December, ok. So, we this month is a 4 bit input because there are 12 months. So, I can have I need 4 bits for encoding this 12 months ok. So, 4 bit month bits are coming. And leap year is a flag so, this leap year if it is it may be 0 it may be 1, ok.

So, this is the final block that we are designing ok. So, here you see that we have got as input this 4 bit month and leap as input. And there are 4 outputs 28, 29, 30 and 31. So, if it is 28 day in that month, then this output will be 1, rest will be 0 similarly the number of days in the month is 29, then this output will be 1, rest will be 0, that way similarly the 30 and 31.

So, if we look into the truth table so, it is like this. So, here the month is 4 bit so, if it is so, our month will start with January. So, we are assuming that we will be entering 1. So, 0 0 0 0 is never given as month so, for that I do not care the output. So, for 0 0 0 0 whatever be the leap year value so, output need not be considered, ok. So, that is do not care. When 0 0 0 1 so, if it is the again the leap year need not be considered leap year is do not care. So, it is 0 0 0 1 so, it is a January so, 31 day month.

Then this 0 0 1 0 is the February, now I have got the leap year, this is without leap year and this is with leap year. So, in with leap year 29 will be a high, and without leap year 28 will be high so, it is goes like this. So, this way I can draw this table, now this one sorry 8 so, this is my month 12 December. So, this is accordingly it is 31 month then. So, this is 1 1 0 1 and 1 1 1 0, 1 1 1 1 so, they are all do not care, because we do not have 13 14 15 months, ok; so, they are all do not cares, this way I can make the truth table.

(Refer Slide Time: 14:45)



Implementation

28 = m8' m4' m2 m1' leap'

29 = m8' m4' m2 m1' leap

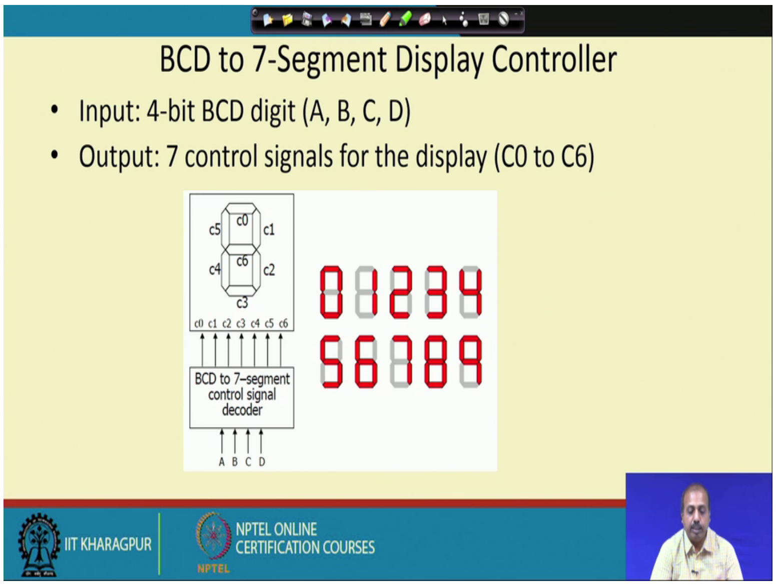
30 = m8' m4 m1' + m8 m1

31 = m8' m1 + m8 m1'

Once you have made the truth table so, we can write down the corresponding combinational function, ok. So, 28 is m 8 dash m that is m 8 bar m 4 bar m 2 m 1 bar leap bar. So, 28 you see that, a 28 is one only for this combination. So, a this all this bits are the m 8 bar so, this is m 8 bar, m 4 bar, m 2 m 1 bar, and leap bar similarly 29 is for this 30 is; so, this is the these are the corresponding logic expression. So, what you can do, for getting this you can draw the corresponding truth table is or from this truth table you can make the corresponding Karnaugh maps and do minimization and get the logic function.

And then in terms of logic gates so, you can realize this combinational function.

(Refer Slide Time: 15:30)



BCD to 7-Segment Display Controller

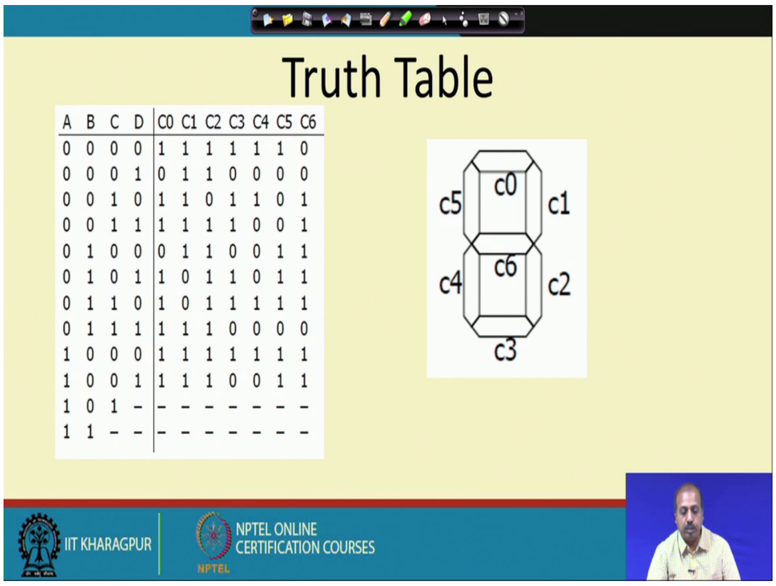
- Input: 4-bit BCD digit (A, B, C, D)
- Output: 7 control signals for the display (C0 to C6)

c0
c1
c2
c3
c4
c5
c6
c0 c1 c2 c3 c4 c5 c6
BCD to 7–segment control signal decoder
A B C D
0 1 2 3 4
5 6 7 8 9
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Next we look into another example which is BCD to 7 segment display controllers. A 7 segment display is quite common so, you can find many such displays on the roads and all. So, here we have got some; so, actually this 7 segments are there. So, each segment is a light emitting diode so, you can a control it to be a high or low. So, it you can turn it on or off so, for that I have to give a one to that segments so, that the segment will be turned on. And for give a making it off so, 0 should be given.

Now, normally we want to display the digits 0 to 9. So, we have got a 4 bit binary coded decimal digit, say and they are the this value is coming here this ABCD is the 4 bit binary BCD value that we are having. Now based on this inputs so, I have to select the appropriate segments. So, this segments are named as C 0 C 1 C 2 C 3 C 4 C 5 C 6 so, 7 segments. So, for example, if you give 0, then excepting C 6 all other segment should be turned on similarly if you are giving 8 as input then all the segments should be turned on.

(Refer Slide Time: 16:40).



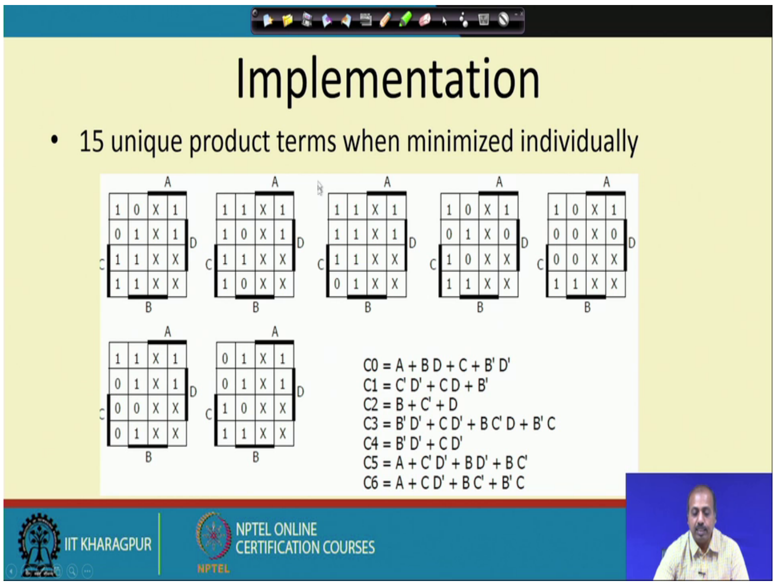
Truth Table

A	B	C	D	C0	C1	C2	C3	C4	C5	C6
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	–	–	–	–	–	–	–	–
1	1	–	–	–	–	–	–	–	–	–

So, I can draw the corresponding truth table like this. So, for ABCD equal to 0 0 0 0. So, C 0 to C 5 all are 1 and C 6 equal to 0. And similarly whenever you have the outputting say for 8 1 1 0 0 0. So, we have got for 1 0 0 0 we have got all inputs or all the segments to be turned on so, we have got all one.

So, for 9 we have got this, for 10 11 12 13 14 15 so, the they are do not care so, do not have to do anything.

(Refer Slide Time: 17:11)



Implementation

- 15 unique product terms when minimized individually

1	0	X	1
0	1	X	1
1	1	X	X
1	1	X	X

1	1	X	1
1	0	X	1
1	1	X	X
1	0	X	X

1	1	X	1
1	1	X	1
1	1	X	X
0	1	X	X

1	0	X	1
0	1	X	0
1	0	X	X
1	1	X	X

1	0	X	1
0	0	X	0
0	0	X	X
1	1	X	X

1	1	X	1
0	1	X	1
0	0	X	X
0	1	X	X

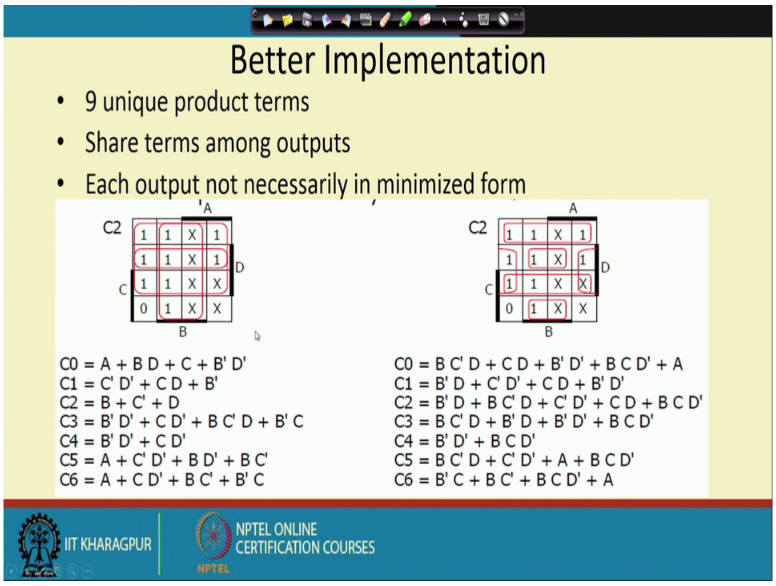
0	1	X	1
0	1	X	1
1	0	X	X
1	1	X	X

$C0 = A + B D + C + B' D'$
$C1 = C' D' + C D + B'$
$C2 = B + C' + D$
$C3 = B' D' + C D' + B C' D + B' C$
$C4 = B' D' + C D'$
$C5 = A + C' D' + B D' + B C'$
$C6 = A + C D' + B C' + B' C$

So, for getting the corresponding implementation, you have to draw the corresponding Karnaugh maps so, these are the Karnaugh maps for the a different conditions, ok. And then so, these are the corresponding Boolean expression; so, C 0 C 1 C 2 C 3 C 4 C 5 C 6 so, these are the Boolean expressions.

And once we have this Boolean expressions so, you can realize them using the logic gates, ok, so, that can be done.

(Refer Slide Time: 17:38)



Better Implementation

- 9 unique product terms
- Share terms among outputs
- Each output not necessarily in minimized form

C0 = A + B D + C + B' D'
C1 = C' D' + C D + B'
C2 = B + C' + D
C3 = B' D' + C D' + B C' D + B' C
C4 = B' D' + C D'
C5 = A + C' D' + B D' + B C'
C6 = A + C D' + B C' + B' C

C0 = B C' D + C D + B' D' + B C D' + A
C1 = B' D + C' D' + C D + B' D'
C2 = B' D + B C' D + C' D' + C D + B C D'
C3 = B C' D + B' D + B' D' + B C D'
C4 = B' D' + B C D'
C5 = B C' D + C' D' + A + B C D'
C6 = B' C + B C' + B C D' + A

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

The better implementation can be like this that there are 9 unique terms ok. So, if you are looking into this functions; see this A is coming here as well as there.

So, this then this say this B bar C. So, B bar C is there in C 3 as well as in C 6, ok. So, in this way if you look into the if you look into the patterns, then many of this product terms are common between the functions.

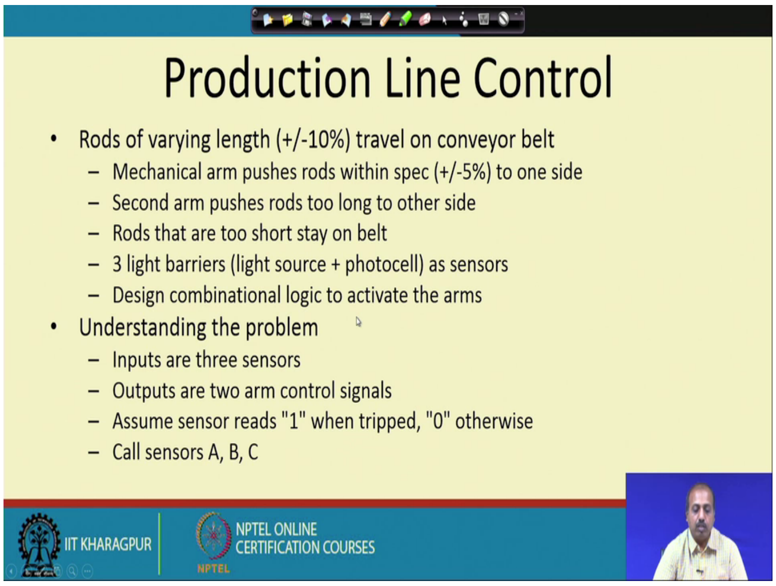
So, to do so, this is the situation; so, this B bar C is common and all. So, what we do, we can modify it so that this is the this commonality can be used, ok. So, this commonality can be used so, that way I will have a less number of terms ok. So, less number of terms will be realized. So, this so this they will share the number of outputs and then the of course, the each output is not in minimized form like you see that here we have got so, this B in this C 0, it was A plus BD plus C plus B bar D bar. But we have modified this A so the C 0 so that the expression becomes like this.

That is we do a grouping so that it is not the optimal one, but what has happened is that this BC bar D is shared between C 0 and C 3. Similarly, this B bar D bar is shared between say this C 0 and a this C 3. So, that way we can make this terms common between this multiple outputs so that total number of unique terms becomes less ok. So, this is actually a combinational logic minimization problem and for multiple outputs this is a very difficult problem in fact. And there are many cad tools that are available which does this type of optimizations, where it tries to share terms between the logic functions to get it the to get the minimum number of product transfer the whole function.

So, whatever minimization technique we have learnt using Karnaugh map and all, so, they are for single output function. So, whenever we have got multi output function, then there is a goal to share between the share the product terms between outputs. And this is a one such example where we see that if we group in a different fashion, then for example, this C 2 instead of grouping like this, where we are getting say B plus C bar plus D.

So, in this type of grouping so, you will get C 2 as a term like this. But the grouping is that you do not need any new term so, for example, this B bar D is there in C 1 also C bar so, the B C bar D is there in C 0 also, C bar D bar is there here also CD is also there, and then BC D bar is also there in C 0. So, as a result more number of terms are getting shared so, using only 9 unique product terms so, you can realize all the functions C 0 to C 6 ok; so, that can be done.

(Refer Slide Time: 20:51).



Production Line Control

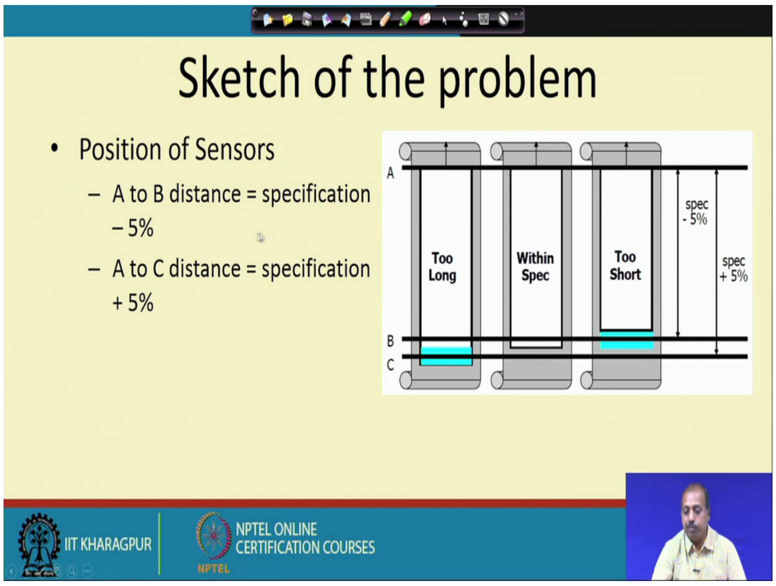
- Rods of varying length (+/-10%) travel on conveyor belt
  - Mechanical arm pushes rods within spec (+/-5%) to one side
  - Second arm pushes rods too long to other side
  - Rods that are too short stay on belt
  - 3 light barriers (light source + photocell) as sensors
  - Design combinational logic to activate the arms
- Understanding the problem
  - Inputs are three sensors
  - Outputs are two arm control signals
  - Assume sensor reads "1" when tripped, "0" otherwise
  - Call sensors A, B, C

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Next we will consider another example which is production line control so, we have got some rods with of varying length that travel on a conveyer belt, ok. So, the length may vary the length of the rod at the production side so, it can vary between plus minus 10 percent. And mechanical arm pushes rods with specification plus minus 5 percent to one side, and the second arm pushes rods too long to the other to the other side.

So, what is so, the rods are produced with an accuracy of plus minus 10 percent, but we want the rods so, we want the rods to be on the to be or of proper specification.

(Refer Slide Time: 21:36)



Sketch of the problem

- Position of Sensors
  - A to B distance = specification – 5%
  - A to C distance = specification + 5%

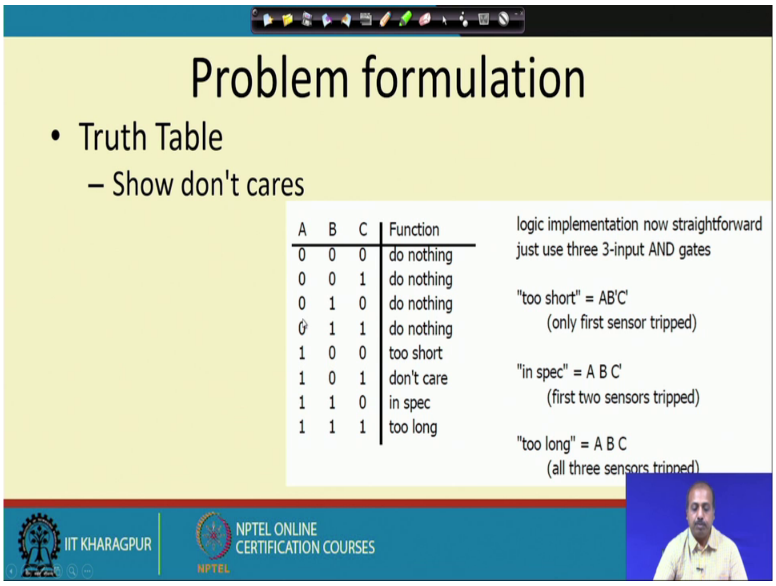
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, we will have these things. So, this is the specification. So, this is within specification the rods are within specification then it will pass.

If the rods are shorter so, that is too short so, they will be taken out by this on this line. And similarly if the rods are larger like here so, it is too long so, then also it will be taken out. So, A to B distance is the specification minus 5 percent so, A to B distance is specification minus 5 percent. So, any rod that passes through this AB so, that is actually not acceptable. Similarly A to C distance is specification plus 5 percent so, that way that is also not acceptable.

So, anything between this plus minus 5 percent is acceptable or, but otherwise it is not.

(Refer Slide Time: 22:31)



Problem formulation

- Truth Table
  - Show don't cares

A	B	C	Function
0	0	0	do nothing
0	0	1	do nothing
0	1	0	do nothing
0	1	1	do nothing
1	0	0	too short
1	0	1	don't care
1	1	0	in spec
1	1	1	too long

logic implementation now straightforward
just use three 3-input AND gates

"too short" = AB'C'
(only first sensor tripped)

"in spec" = A B C'
(first two sensors tripped)

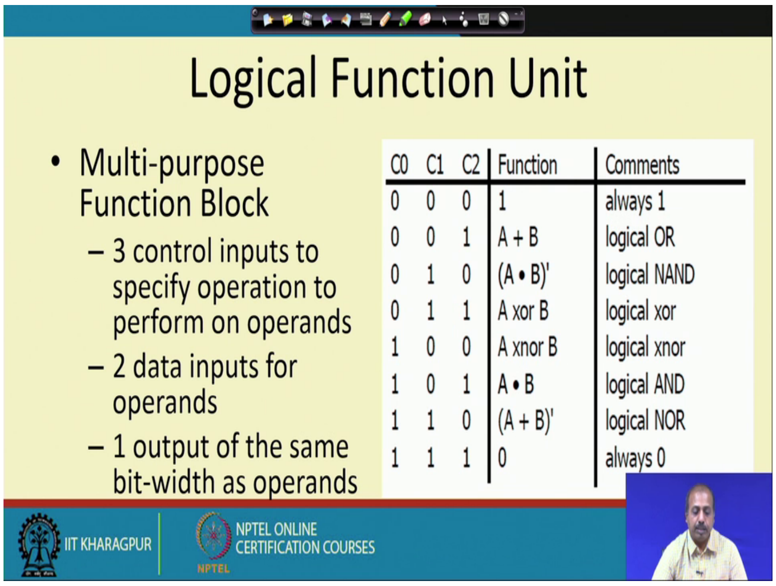
"too long" = A B C
(all three sensors tripped)

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, for this one if ABC is 0 0 0 so, that is basically this ABC so, they are so, it is not violating any of the conditions, then we do nothing. If it is 0 0 1 that is it is not way it this A and B are 0 0 and C is 1 so, then also it is do nothing. So, only when this A is if A is equal to 1 in that case the rod is too short so, output is too short. And then you if it is 1 0 1 is again do not care or do nothing, because that is do not care condition. Then 1 1 0 this is the valid within specification and 1 1 1 is too long.

So, I can write down the corresponding logic expressions for too short in within specification and too long, and accordingly we can get the corresponding function. So, I can have a combinational logic so, based on this ABC outputs from this rail, so, from this rail so, we can identify whether the rod is within specification too long or too short.

(Refer Slide Time: 23:41)



Logical Function Unit

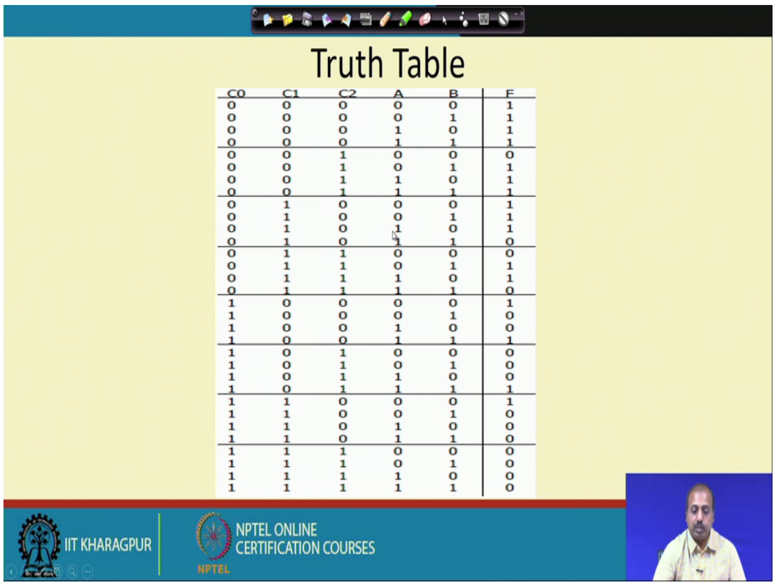
- Multi-purpose Function Block
  - 3 control inputs to specify operation to perform on operands
  - 2 data inputs for operands
  - 1 output of the same bit-width as operands

C0	C1	C2	Function	Comments
0	0	0	1	always 1
0	0	1	A + B	logical OR
0	1	0	(A • B)'	logical NAND
0	1	1	A xor B	logical xor
1	0	0	A xnor B	logical xnor
1	0	1	A • B	logical AND
1	1	0	(A + B)'	logical NOR
1	1	1	0	always 0

Another example so, we have got a multipurpose function block. So, this is that is a C 0 C 1 C 2 so, these are the control inputs, and there are 2 data inputs A and B. So, based on these C 0 C 1 C 2 so, it will perform some function.

So, for example, if C 0 C 1 C 2 is all of them are 0 so, it is a constant function. So, irrespective of the values of A and B o, it will output A 1. If C 0 C 1 C 2 is 0 0 1 then it will compute A plus B. So, it is a logical or of A and B so, 0 1 0 is logical NAND, 0 1 1 is logical XOR. So, like that we have got different functions to be implemented if the C 0 C 1 C 2 are having some specific value.

(Refer Slide Time: 24:25)



Truth Table

C0	C1	C2	A	B	F
0	0	0	0	0	1
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Now, so, this is the truth table so, this C 0 C 1 C 2 so, this is 0 0 0, then this A and B they are having so, this is a constant function. So, irrespective of the values of A and B output will be 1. Similarly this for 0 0 1 so, this is the OR of A and B so, this is the OR function is realized here. Then this 0 1 0 so, this is the NAND function so, this doing a NAND here. So, this way you can draw the truth table.

And after you have made this truth table you can do a logic minimization to get an expression for F in terms of C 0 C 1 C 2 A and B ok, and that can give us the that can give us the expression. So, that can give so that the a final circuit is not shown here, but you can always do some logic minimization and get the function the logic the digital circuit implemented in terms of C 0 C 1 C 2 A and B.