

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

Lecture – 35
Time Closure (Part 4)

So, welcome back, what you have seen in our previous lectures is that how we can carry out some timing analysis, the complexity of which is linear in the size of this circuit and hence can be used for very large circuits to enumerate the paths, the nets, so as to adjust the slack values depending on the set up and whole time constrains.

Now one thing I mentioned earlier, but we did not say how to handle it, you see given a circuit we are just looking at the topological net list and try to find out all the paths, try to calculate this slacks adjust them to zeros using this zero slack algorithm and so on. But one thing we did not consider yet is that the paths that we are considering are they all actual important to consider; like there may be some paths which may look to be very critical very long, but what may it happening in practice is that this circuits are such that because the functionality of the blocks gates whatever that is important. So, what kind of signal can go through? So, it is possible that the gates or the blocks are such that signal transitions can never flow through that long path, which means whatever timing analysis were carrying out on that path that is not important in the context of the present functionality or the design. So, there is a concept called false path that we shall be particularly considering in this lecture.

(Refer Slide Time: 02:13)

False Paths

- Paths that physically exist in a design but are not logic / functional paths.
- These paths never get sensitized under any input conditions.
- An example is shown on the next slide.

So, by false path as I said we refer to paths in the circuit net list that physically exist, but are not logic or functional paths like in your circuit if you consider this circuit net list or the dag, as a directed acyclic graph there will be a path from point to another, but looking at the functionality of the nodes or the blocks that path will never cause a flow of signal because of the property of this blocks, which means there is a (Refer Time: 02:56) sensitization we shall be looking at it in detail more detail, this paths never get sensitized for any combination of the inputs.

(Refer Slide Time: 03:08)

An example:
The path of length 400 is never exercised.

So, let us take an example what do mean by this. Let us take simple example like this, what is there? There are 2 multiplexors; these are the multiplexor there is a select input. So, if s is 0 this 0 input is selected, if s is 1 the 1 input is selected and these rectangular boxes are some circuit modules, so we are not showing the detail but the delay unit is shown

This block has 200 units of delay this can be combination or circuit this can be another combination circuit 100, 200, 100. So, if you carry out static timing analysis of this circuit, then the longest path that s t a will give you is probably this 200 well assuming MUX delay is negligible, 200 plus 200, 400 right. So, this 400 will be the longest path that s t a will give you and you are most likely this s t a and this subsequent corrective steps you have to spend maximum amount of time in adjusting this path, so that this slacks are made 0 and so on and so forth right, but if you look at the functionality there are multiplexors, you know how multiplexor works.

So, if you look at the way the select lines are activated. So, if s equal to 0 then v is selected and on this side if there is an inversion s is 0 mean, this becomes 1. So, x is selected. So, the delay will be 100 plus 200, 300. Now if s equal to 1 then u is selected and this becomes 0, so this y is selected. So, now, signal flows through this path again delays 300. So, this longest path which apparently will show up in your direct a cyclic graph of 400 is never exercised, this is termed as a false path. So, false path can be identified by analyzing the functionality of these blocks or the modules that are there in the circuit. So, just by looking at the topology just in the graph, you cannot tell which path is false or which path is not false, but if you also look into the functionality of the gates of the blocks in addition, you can possibly identify some of the path that can never be exited, right.

So, in this case for example, you can ignore the 400 length path, and you can only consider the other paths. Now another motivation behind having this apriory step of detecting false path and removing them from the consideration is that, there are many circuits where number of paths can grow exponentially with the number of gates or in number of lines well. One classical example is a multiplier. So, for those circuits as this size of the circuit grows, number of paths becomes exponentially large particularly for circuits with the large number of (Refer Time: 06:34) or exclusive or gates, such case do happen. So, for those circuits in particular this false path identification becomes very

important, otherwise the complexity of the whole thing also becomes linear exponentially because number of paths or also exponential, fine.

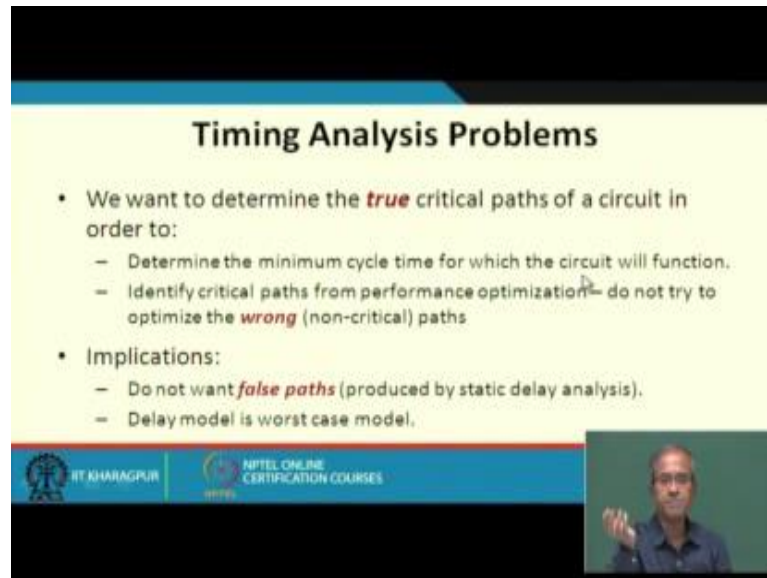
(Refer Slide Time: 06:53)

The slide is titled "Multi-cycle Paths" and contains a bullet point: "Data paths that require more than one clock period for execution." Below the text is a circuit diagram showing two flip-flops connected by a combinational logic block. A double-headed arrow above the logic block is labeled "2 clock period delay". The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

Another situation may be you can have something called multi cycle paths, which means there is some combinational logic between flip flops, but as a designer you know that this will require a 2 clock cycle delay. So, the data value that to be stored here that will take meaningful interpretation only after 2 clock pulses. So, you will not be reading out the output of this flip flop, before 2 clocks wise; so some design may be like that. So, if you have criteria like this, then for this combinational part of the circuit you need not constrain the delay to be within that clock period of t .

So, again if you only look at the topology of this circuit you will not be able to get this I means idea or information, you will additionally have to know from the user that there are some paths which are suppose to be multi cycling nature, and they have to handled with relaxed timing constrains fine.

(Refer Slide Time: 08:08)



The slide is titled "Timing Analysis Problems" and contains the following content:

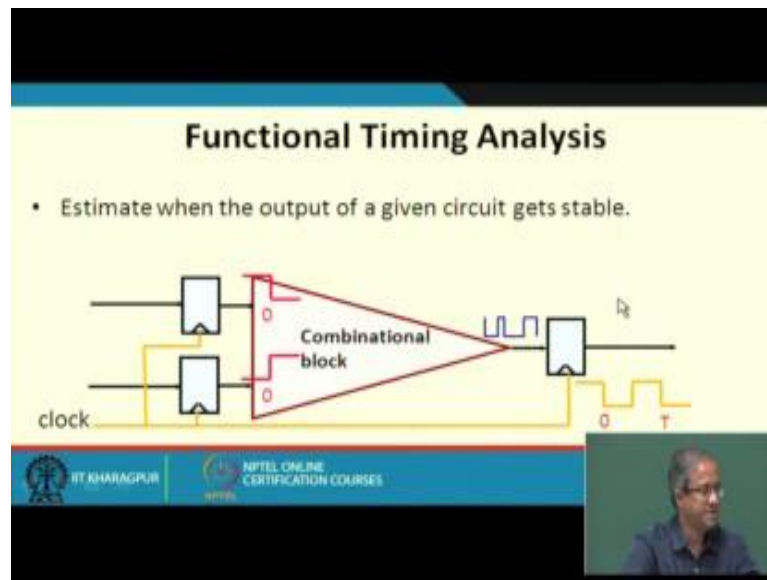
- We want to determine the **true** critical paths of a circuit in order to:
 - Determine the minimum cycle time for which the circuit will function.
 - Identify critical paths from performance optimization— do not try to optimize the **wrong** (non-critical) paths
- Implications:
 - Do not want **false paths** (produced by static delay analysis).
 - Delay model is worst case model.

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset in the bottom right corner shows a man in a blue shirt gesturing while speaking.

So, talking about the timing analysis problem, here we are trying to identify true and false critical paths. So, actually what you want is that we want to determine the true critical paths, why? Because the true critical paths will actually determine the minimum cycle time for which the circuit will function, you think of that multiplexor example I gave, a circuit apparently shows a path of delay 400, but that 400 will never come. So, you never design your clock period taking that 400 into consideration, you take the 300 as the maximum delay of the combination network and you design your clock frequency or period accordingly right fine.

So, this is one and the second thing is I means you normally have to identify the critical paths by looking at this slag values, the slag value if they come to be negative those lines have not as critical. So, you do not unnecessarily try to mark the wrong paths as critical and just unnecessarily put some effort in optimizing them right. So, the implication is that you one false paths to be identified and you do not want unnecessary manipulation or computation of false paths, that are produced by static delay analysis like static timing analyzer where the delay model is the worst case delay model right, this is problem with timing analysis. So, just you look at there is something called functional timing analysis. So, you assume that this is a combinational circuit block.

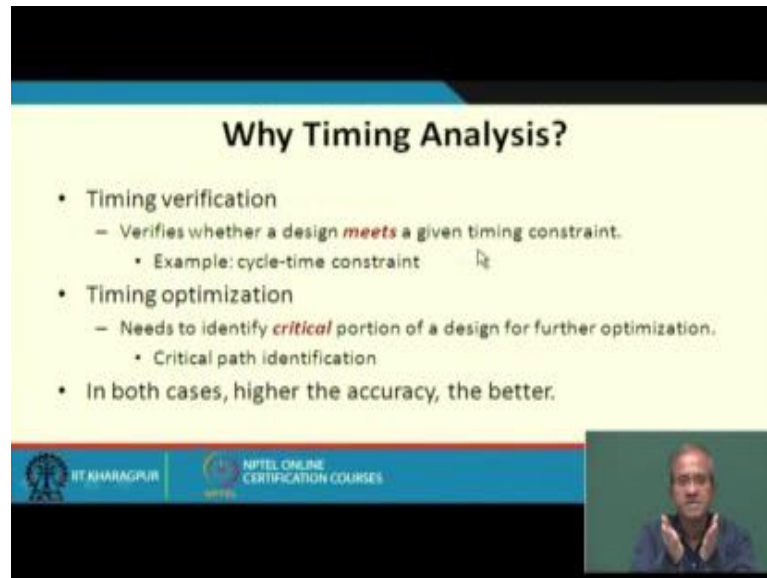
(Refer Slide Time: 10:01)



These are fed from some data coming from flip flops here, and the output is also going in the flip flop here. The clock is feeding all the flip flops, it is of a period T ; now you see suppose that after the clock comes here there is no skew assume, that the inputs with the combinational circuit there making transitions all at time T equal to 0, 0 is a place where the input transitions are taking place.

Now inside the combinational circuit there can be a multiple paths because of the unequal delays, suppose the output is changing before they are getting stabilized. So, you have to analyze the combinational block and find out what is the maximum delay you have to spend after which the output gets stabilized; so that when the next clock is comes, the stable input value should be available here of course, taking into account the set up time constraint as well, right.

(Refer Slide Time: 11:38)



The slide is titled "Why Timing Analysis?" and is presented in a video lecture format. It features a yellow background with black text. The content is organized into a list of bullet points. At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL Online Certification Courses. A small inset video of a man speaking is visible in the bottom right corner of the slide area.

Why Timing Analysis?

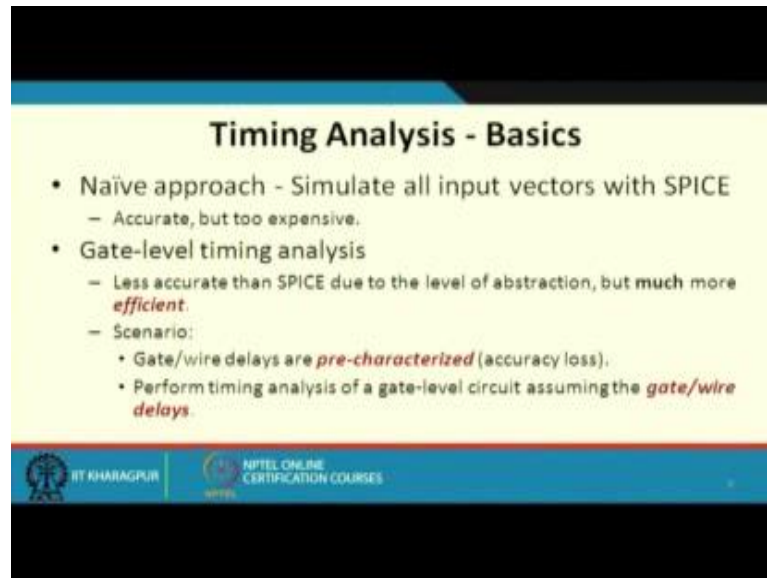
- Timing verification
 - Verifies whether a design *meets* a given timing constraint.
 - Example: cycle-time constraint
- Timing optimization
 - Needs to identify *critical* portion of a design for further optimization.
 - Critical path identification
- In both cases, higher the accuracy, the better.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

This maximum words is delay analyzes is normally done in functional timing analyzes. So, in static timing analyzer also you do something similar right. So, in timing analysis you basically verify whether some design meets a given timing constraint like in this static timing analysis also your doing the same thing, given a net list given the required arrival times are it is you see whether some slag values are negative or not. If it is negative it means your design does not meet this specification, you have to make some adjustments. So, this is the main objective of timing analysis. So, when you are doing timing optimization as a subsequent step to timing analyzes, you are identifying the critical portions of your design like for example, the portions where the slacks are negative.

Then once you identify the critical portions to carry out certain optimization on those. So, once you do that then all your timing constrains will be met. So, both for verification and optimization steps, if you can achieve higher accuracy to be better because you will get a solution which we need to be means again check the modifier at a little stage. So, if you get accurate estimates at earlier stages, then you can have design which will sustain throughout the design cycle. So, you need not have to change it.

(Refer Slide Time: 13:09)



Timing Analysis - Basics

- Naïve approach - Simulate all input vectors with SPICE
 - Accurate, but too expensive.
- Gate-level timing analysis
 - Less accurate than SPICE due to the level of abstraction, but much more *efficient*.
 - Scenario:
 - Gate/wire delays are *pre-characterized* (accuracy loss).
 - Perform timing analysis of a gate-level circuit assuming the *gate/wire delays*.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the basic concept of timing analysis is means you can have a so called naïve approach like you know, but spice is a circuit simulation tool which is pretty accurate. Spice take some very accurate and will establish model of very circuit components like a transistor, resistance capacitance inductance are their also it can model the transmission lines interconnections there are various models available.

So, basically you translate your design or circuit net list in to an equivalent circuit, which consists of voltage source, current sources controlled voltage sources controlled current sources R L C values etcetera and subsequently you carry out a traditional circuit analysis, which typically requires lot of computation, but the advantage is that if your circuit model is good enough the results you get will be very accurate and will quite closely tally with the real life values which you get out of the fabricated circuits. But the trouble with circuit simulation is that as it is slow, it I mean it carries out lot of computation you cannot scale up the simulation for larger circuits, you can do it only for smaller size circuits. So, spice simulation is accurate, but too expensive for larger circuits. So, you may have to go for gate level timing analysis which is compromise.

This of course, will be less accurate than spice, but it will be more efficient, you can run it for say may say is million get circuit for example. So, what you do here is that just like your timing analysis whatever you had seen earlier, your gate or wide delays are pre characterized. You have a simplified model that you can estimate that this is the delay of

my gate; this is the delay my wire, they ways to estimate without going through and elaborates calculation or computation. So, using that you can make some simple delay estimates, and using those you can carry out a timing analysis, that is actually what is saw earlier when we discussed the static timing analysis and also the 0 skew algorithms right, fine.

(Refer Slide Time: 15:43)

Gate-level Timing Analysis

False path aware
arr(z)?

z |
1
1

X₁ X₂

arr(x₁)=0 arr(x₂)=0

- A naive approach is topological analysis.
 - Easy longest-path problem
 - Linear in the size of a network
- Not all paths can propagate signal events.
 - False paths
 - If all longest paths are false, topological analysis gives delay overestimate.

Functional timing analysis = false-path-aware timing analysis

- Compute false-path-aware arrival time

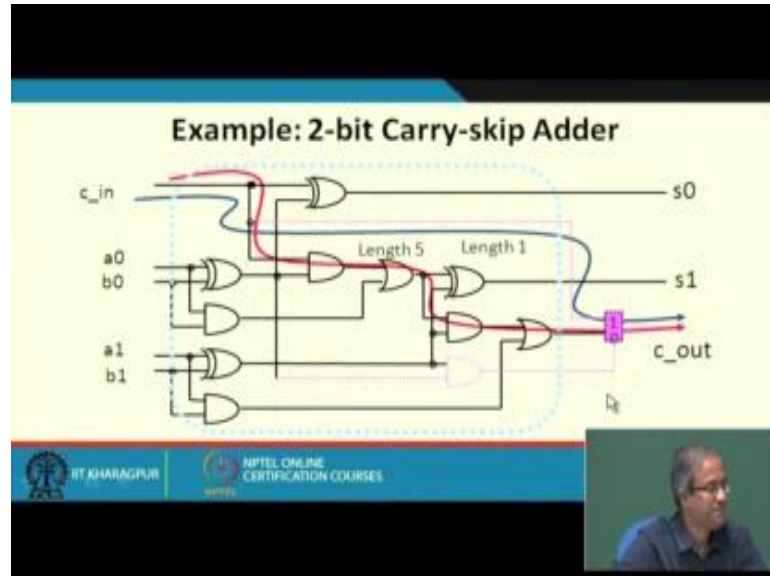
IT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, just look at this example. So, if you are doing a gate level timing analysis, there can be some problems you look at this circuit, what this circuit realizes? That to end gates, the output of this end gate is just the end of X 1 and X 2 that your ending with X 2. So, this Z is also X 1 and X 2 and of X 1 X 2. So, you can say that this end gate is actually this path, this path is reddened end. So, you can have this X1 directly here, X 1 and X 2 is X 2. So, gate level timing analysis is again a naïve approach because you determine with an longest path without looking at the functionality. So, what I am saying here is that not all paths can propagate signal events. So, no signal event would be propagated along this path, there will be propagate along this path and along this path.

So, this path which was the longest path, so if it is false then whatever estimate you get from the gate level analysis will give you an over estimate, it will give you longer delays. So, whenever we are doing functional timing analysis, you should be doing false path aware timing analysis; that means, you identify the false path and accordingly you compute the arrival time, so that you get a good estimate and Z. Because if you just look

at the topology and just do it, you will be typically getting a higher value for the arrival times, because you are also considering the false paths.

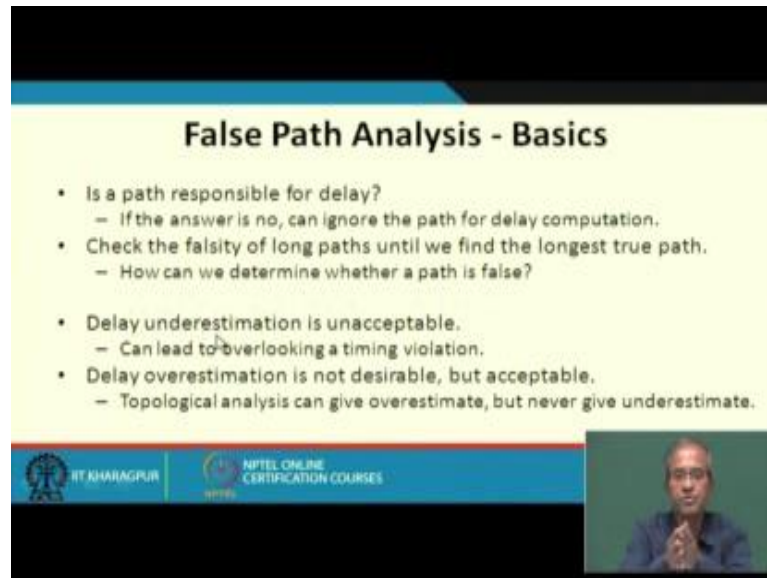
(Refer Slide Time: 17:33)



Take an example of a 2 bit carry skip adder, just a simple example you see the net list where 2 pairs of input a_0, b_0 and a_1, b_1 are there and some s_0, s_1 and the carry out is given.

Now, what I am saying is that you can have a multiplex are here possibly as an addition, where you can take the carry in and you can add getting circuit where you take the x or of a 1 b 1 and take the this input also which is x or of a 1 b 1 and x or of a 0, b 0 end it and fill it as select lane, which means this indicates the carry propagate condition. So, if there is a carry propagate condition then whatever is in seen that will propagate to the output. So, if you do this, then you are carrying the propagating along this line, but if you just calculate the longest path, longest path is this, but by introducing this we have introduced a case, where actually for this longest path scenario will carry will be moving through this fast path through this, but using this topological analysis you cannot identify that that is the basic idea.

(Refer Slide Time: 18:59)



The slide is titled "False Path Analysis - Basics" and contains the following bullet points:

- Is a path responsible for delay?
 - If the answer is no, can ignore the path for delay computation.
- Check the falsity of long paths until we find the longest true path.
 - How can we determine whether a path is false?
- Delay underestimation is unacceptable.
 - Can lead to overlooking a timing violation.
- Delay overestimation is not desirable, but acceptable.
 - Topological analysis can give overestimate, but never give underestimate.

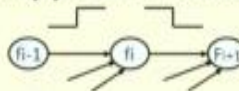
The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

So, let us look in to the false path analysis now some of the basic concepts. So, you try to check whether the path is actually responsible for the delay from an input to given output or not. If it is no just ignore that path, that path is not required to be considered. So, you check whether long paths correspond to false path or not, you may see that the whole path is not false, but at least a part of it is false. There are ways of identifying whether the path is false or not; we shall be looking at a couple of strategies later. Now you see delay underestimation false path what are you doing, we are sometimes saying that we actually delay we are calculating is not correct the actual delay should be less. Some paths through which the single is propagating is false, but some nave methods for detecting false path may say that well I have a shorter path, my delay should be this, but that longer path can also play a role under some cases.

Like you are means effectively making an underestimation of the delay, what whenever you are making an underestimation of a delay there may be a problem, because you might be ignoring some timing violation; on the other hand delay over estimation is not desirable, but if in some cases they do exist you can accept them also that will not give you an incorrect result, fine.


(Refer Slide Time: 20:46)

Possible Approach :: Boolean Difference



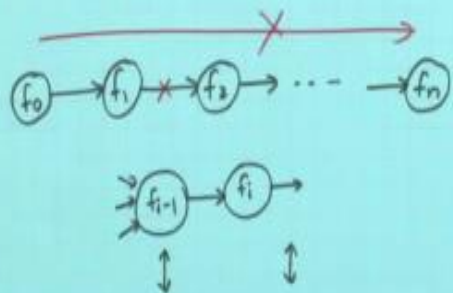
- Path $P = \{f_0, f_1, f_2, \dots, f_n\}$
 $\frac{\partial f_i}{\partial f_{i-1}}$ gives conditions under which node f_i is "sensitive" to node f_{i-1} .
- So output P is sensitive to f_0 if $\prod_{i=1}^n \frac{\partial f_i}{\partial f_{i-1}} = 0$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES




So, here one simple method we present here this is based on a concept called Boolean difference. See you consider a path consider f_0, f_1, f_2, f_n these are the nodes the idea is like this you start with f_0 there will be f_1, f_2 up to f_n these nodes may indicate gates or any functional blocks.

(Refer Slide Time: 21:07)



11.07.2017



So, this Boolean difference what it says is that it says let us take in general a block f_i . So, it is getting its input from f_{i-1} . So, what Boolean difference says the questions like this says that if there is a signal transition at f_{i-1} , then is it possible to get a

signal transition at the output of f_i yes or no. If it finds out that well there exists some combination of inputs for which this signal transition can propagate then it is fine, but along this path if you find that there are some pairs of f_i minus 1 f_i let us say here, for this signal cannot propagate which means this entire path is a false path, this signal cannot propagate through this. So, across every such blocks pair wise if you compute the Boolean difference and if you cannot establish that a transition here can propagate here, you can say that this path is not false in path is false you cannot propagate.

So, the Boolean difference is actually denoted like this between block f_i and f_i minus 1, you use it in the del notation derivative notation Δf_i ; Δf_i means if there is a change in f_i minus 1 can there be a change in f_i ? So, this gives logic expression Boolean expression. So, if you solve it I means you can get either or null function equal to 0, which means it is no possible or you can get an expression in terms of input variables which tells you that for which input combinations this is possible.

(Refer Slide Time: 23:32)

Example :: Static False Path

$f_i = su + \bar{s}v$

$f_j = \bar{s}x + sy$

$\frac{\partial f_i}{\partial u} = (s + \bar{s})(s + \bar{s}) + \bar{s}(\bar{s}) = s$

$\frac{\partial f_j}{\partial v} = \bar{s}$

Hence, $\frac{\partial f_i}{\partial u} \cdot \frac{\partial f_j}{\partial v} = 0$

The path is not sensitizable and hence is false.

IIT KHARAGPUR
 NPTEL ONLINE CERTIFICATION COURSES

So, the final output will be sensible will be sensitive to change in f_0 , if all of these Δf_i Δf_i in values give non null values, that if you take the product the final product will be non null this is the basic idea. So, I will like just take a simple example, we will take that same example of a multiplexor. So, let us take f_i . So, f_i the output expression u and v are the inputs and s is the select. So, if s is 0 v , if s is 1 u . So, I can write the output expression like this su plus $\bar{s}v$. Similarly f_j , if s is 0 then this is 1 the next

otherwise. So, f_j I can write like this \bar{x} plus $s y$. If I calculate $\frac{\partial f_i}{\partial u}$, $\frac{\partial f_i}{\partial u} \frac{\partial u}{\partial f_i} \frac{\partial f_i}{\partial x}$ means I one expression like this, the definition of a Boolean difference is like this. So, I am just noting down for any function f .

(Refer Slide Time: 24:23)

$$\frac{\partial f}{\partial x} = \underbrace{f(x=0) \oplus f(x=1)}_{\text{Boolean Difference}}$$

So, if I say Boolean difference is $\frac{\partial f}{\partial x}$ del some input variables x , there are many input inputs one of them is x . This means the value of the function when x equal to 0, exclusive or the value of the function when x equal to 1.

This means if x changes does the output of f changes? This expression actually tells you that that how sensitive f is with respect to x . This is the definition of Boolean difference. So, let us see here. So, here $\frac{\partial f_i}{\partial u}$ will be if you just uses that formula you just now shown and if you do a simplification, you can find $\frac{\partial f_i}{\partial u}$ comes to be only s . Similarly for $\frac{\partial f_j}{\partial x}$ comes to \bar{s} . So, we are actually talking about this path $u \rightarrow f_i$, then $y \rightarrow x \rightarrow f_j$ and if I do product of this 2 functions we see that $s \cdot \bar{s}$ cancels out become 0, this means this is a false path, signal cannot propagate from u to f_i and f_i to x to f_j together.

This is a simple example how Boolean difference can be used to identify a false path. So, the path is not sensitizable and is false.

(Refer Slide Time: 26:09)

Definitions

- Given a simple gate (i.e. AND, OR, NAND, NOR), a **controlling value** on an input determines the output of the gate independent of the other inputs.
- Given a simple gate (i.e. AND, OR, NAND, NOR), a **non-controlling value** on an input cannot determine the output of the gate independent of the other inputs.
 - 0 is a controlling value for AND gate; 1 is non-controlling value for AND gate.
- Controlling/ non-controlling value is merely a specialization of the Boolean difference to simple gates.

BIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, we shall be using some concepts in the next method that you shall be looking at. So, in this lecture just let us introduce just few concepts and definitions. For every simple gate we define something called a controlling value and a non controlling value. Controlling value means if this value is applied on one of the inputs, it will uniquely determine the output. Non controlling value means not that way, see controlling value and input determines the output of the gates independent of the other inputs just take an example.

(Refer Slide Time: 26:53)

0

x
 x

1

x
 x

a
 $b=1$

$f = ab$

$\frac{\partial f}{\partial a} = f(a=0) \oplus f(a=1)$
 $= 0 \oplus b = \underline{\underline{b}}$

a
 $b=0$

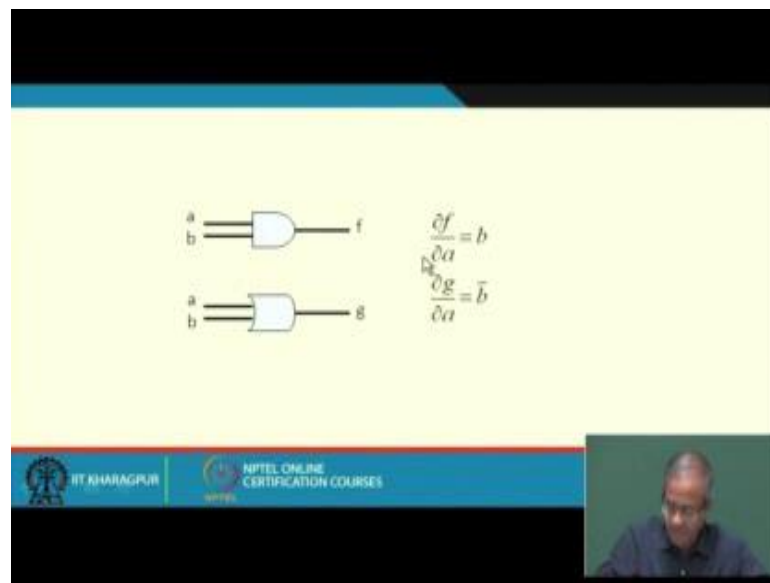
f

$\frac{\partial f}{\partial a} = \bar{b}$

Suppose I have a 3 input AND gate. So, we consider this input. So, if I apply as 0 to this input, then what will be output? Irrespective of the other 2 inputs the output will be 0.

So, here 0 is said to be the controlling input, but if I apply a 1, then I cannot say with the output is the output will be dependent on the other inputs that is a non controlling value. Similarly for an OR gate let us say again this is an input or gate. So, if I apply a 1 then the output will be 1 irrespective of the others, this is a controlling value for an OR right, this is the concept beyond controlling and non controlling value.

(Refer Slide Time: 27:44)

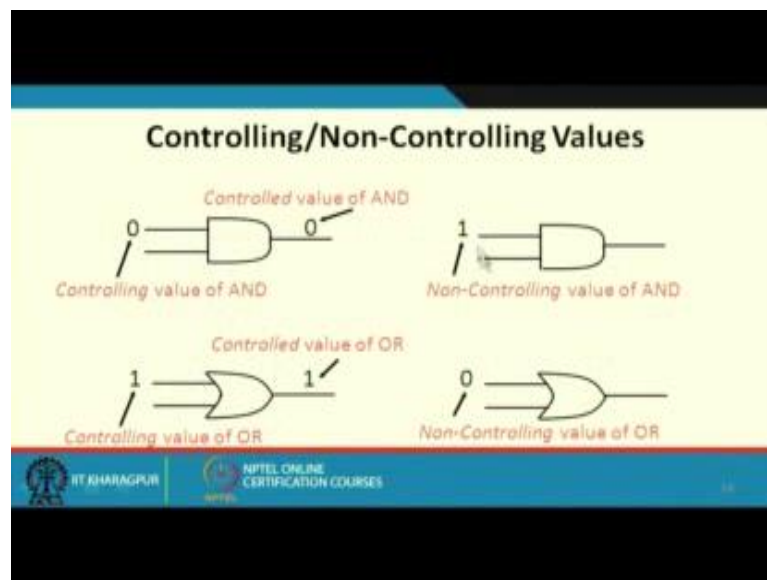


So, this also follows from the Boolean difference concept, you see a b. So, this is end. So, what will be $\frac{\partial f}{\partial a}$? Let us make a calculation say 2 input AND gate a b, the output is f equal to a b. So, if I calculate the Boolean difference of f with respect to a. So, you just recall the definition the function where x equal to 0, a equal to 0 XOR the function when a equal to 1.

So, when a equal to 0 the whole function is 0, XOR if a equal to 1 the function is b, 0 XOR b is b. So, $\frac{\partial f}{\partial a}$ is b. $\frac{\partial g}{\partial a}$ is \bar{b} . So, this actually gives you the condition for sensitivity. So, $\frac{\partial f}{\partial a}$ equal to b means what; $\frac{\partial f}{\partial a}$ equal to b means that if I want to that a is varying right. So, from a to b what is the condition of signal propagation, then this condition has to be holding true; that means, b must be equal to 1, this function has to true. So, if b is 1 then only any signal transition in a will be propagating here. Similarly for an OR gate, so if you do a similar calculation, you will

find that $\text{del } f \text{ del } a$ will be b bar, which means for this case if you want to propagate the signal your b bar has to be true that mean b has to be 0. This gives you the condition for signal propagation, this we shall be utilizing in our next method that we will be discussing later.

(Refer Slide Time: 29:48)



So, just to summarize in this diagram I am showing for the basic gates the controlling and the non controlling values - for AND, 0 is the controlling value 1 is the non controlling value; for OR, 1 is the controlling, 0 is in non controlling, similar is for NAND. For NAND gate also 0 will be controlling, for NOR gate also 1 will be controlling. So, just remember this, but because in our next lecture we shall be looking at a method, where we may have to sensitize a signal across gates we need this concepts. So, with this we come to the end of this lecture.

Thank you.