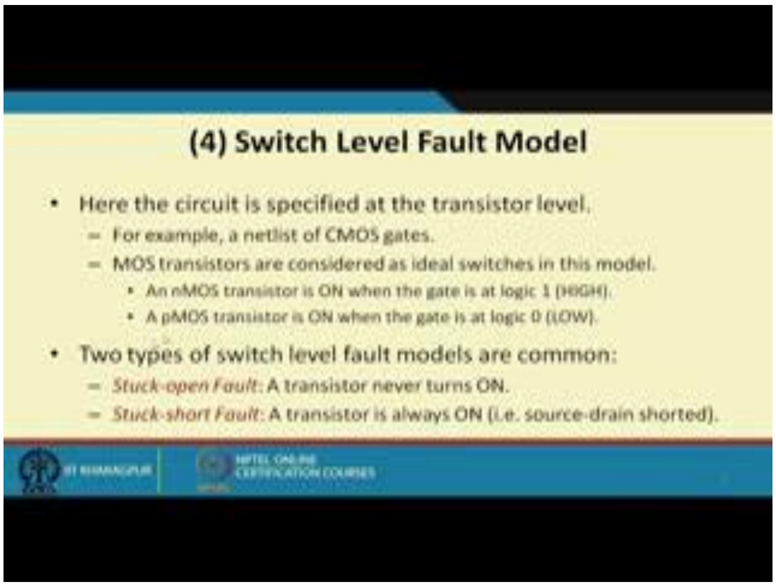
VLSI Physical Design

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

Lecture – 50
Fault Modeling (Part 2)

So, we continue with our discussion on Fault Modeling. So, you recall in your last lecture we talked about some fault model at the behavior level, function level and also the structure level. We mentioned at this structure level this static fault models at the most commonly used, and also bridging fault model is quite common in modern (Refer Time: 00:47) chips where the wires are very close to together.

(Refer Slide Time: 00:51)



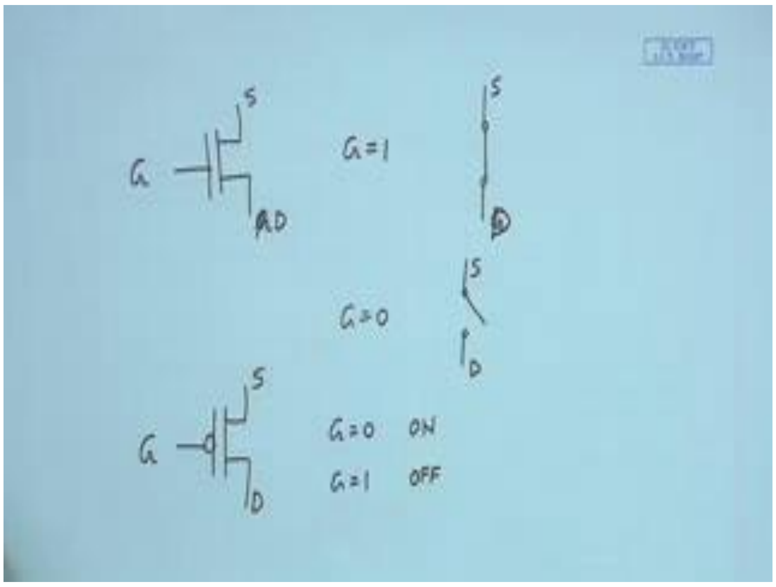
(4) Switch Level Fault Model

- Here the circuit is specified at the transistor level.
  - For example, a netlist of CMOS gates.
  - MOS transistors are considered as ideal switches in this model.
    - An nMOS transistor is ON when the gate is at logic 1 (HIGH).
    - A pMOS transistor is ON when the gate is at logic 0 (LOW).
- Two types of switch level fault models are common:
  - *Stuck-open Fault*: A transistor never turns ON.
  - *Stuck-short Fault*: A transistor is always ON (i.e. source-drain shorted).

So, to continue with the discussion, we first look at today some switch level fault models; switch level means at the level of the transistors.

So, why we call it as a switch level? Because here we regard a transistor MOS transistor as an ideal switch like what I am saying is that, suppose I have an n MOS transistor like this.

(Refer Slide Time: 01:14)

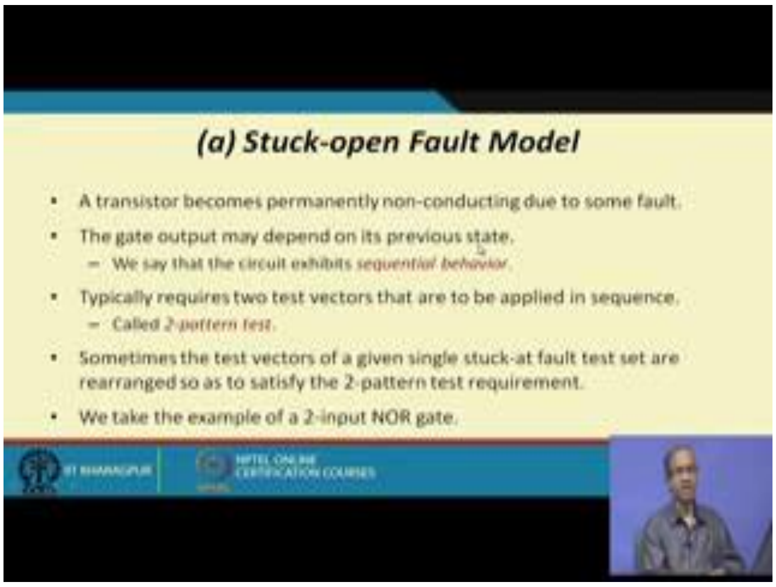


Let us say S and G and this is sorry S and D source and drain and this is the gate, and if I apply gate voltage equal to high then assume this is a switch which is getting connected there is a connection, this is S and G and D, sorry and if G is 0 then the switch will be disconnected S and D it will be disconnected.

Now, if it is a pMOS transistor it is just the reverse logic. If it is a pMOS transistor then if G is 0, then the switch is on if G is 1 then switch is off. So, in this fault model we are assuming that the MOS transistor that comprises cMOS gate, behave like ideal switches where sources and drains are either connected or they are disconnected fine. So, just as I said an nMOS transistor is on when the gate is high pMOS transistor is on when the gate is low.

Now, here we are considering 2 different fault models, first is transistors truck open fault which means the switch is never turning on, source and drain are never getting connected irrespective of the fault age we apply to gate. Second one is reverse is the short, the source and the drains are already shorted. So, it is always conducting irrespective of the voltage you apply to the gate. Now the reason we consider these switch level for separate that we will see that the behavior of the circuit becomes very peculiar in presence of this faults; like other faults models we looked at like stuck at fault model, which is said that is very popular, but stuck at fault model cannot capture some of these transistor level fault models, which modify the circuit behaviors in a very peculiar way that we shall see shortly.

(Refer Slide Time: 03:39)



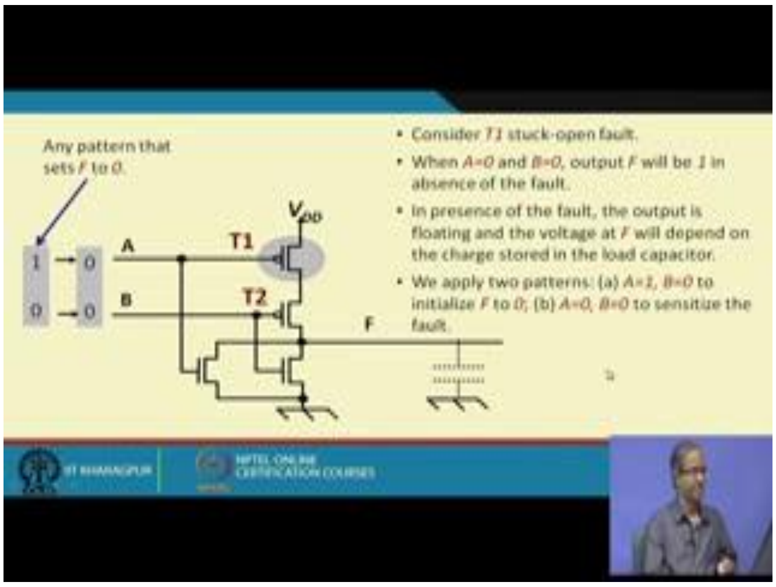
(a) Stuck-open Fault Model

- A transistor becomes permanently non-conducting due to some fault.
- The gate output may depend on its previous state.
  - We say that the circuit exhibits *sequential behavior*.
- Typically requires two test vectors that are to be applied in sequence.
  - Called *2-pattern test*.
- Sometimes the test vectors of a given single stuck-at fault test set are rearranged so as to satisfy the 2-pattern test requirement.
- We take the example of a 2-input NOR gate.

So, look at this stuck open fault model first; here as I said a transistor becomes permanently non conducting, because of some fault. Now we shall see that because of this fault when I apply some input, the gate output may remain the same as it is was in its previous state when the last input was applied, what does this mean that I have a let us say a combination circuit and nand gate or a nor gate, normally for a combination circuit when we apply some inputs, we get some outputs we just a function of the inputs. But now we are saying that I apply a input and the output whatever I am expected to get will not depend on this input, but will depend on the previous input, which means now as if my combination circuit is behaving like a sequential circuit, it is remembering the previous state in some way, right.

So, we say that the circuit exhibits sequential behavior and because of this reason such faults cannot be detected by a single test, we usually require a fair of test vectors which is sometimes called 2 pattern test. So, we shall see with an example how it is done or it is happening. But this 2 pattern test sometimes in practice it is done, that we first generate the setup test vectors for single stuck at faults then we rearrange this test vectors, so that this 2 pattern test requirements are satisfied.

(Refer Slide Time: 05:35)


- Consider $T1$ stuck-open fault.
- When $A=0$ and $B=0$, output F will be 1 in absence of the fault.
- In presence of the fault, the output is floating and the voltage at F will depend on the charge stored in the load capacitor.
- We apply two patterns: (a) $A=1$, $B=0$ to initialize F to 0; (b) $A=0$, $B=0$ to sensitize the fault.

Any pattern that sets F to 0.
1
0
0
0
A
B
T1
T2
V_{DD}
F
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

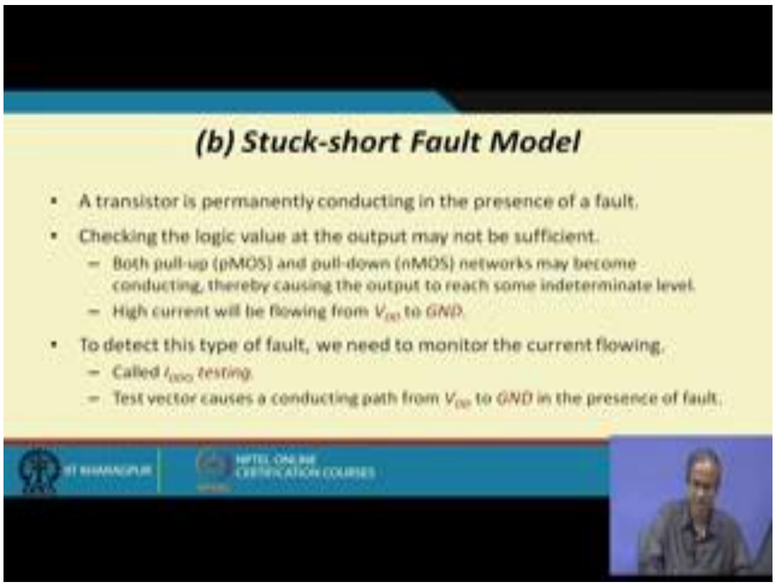
So, let us take the example of a 2 input CMOS NOR gate, and let us illustrate what we are talking about. Here we have a 2 input NOR gate, there are 2 nMOS transistors in the pull down and 2 pMOS transistors here. The output of this gate is driving some other gate or gates; here I am showing the equivalent load capacitance. So, I told earlier that whenever a transistors is driving another transistor or another gate, so the gate capacitance that the this line is facing that is quite significant, and it is that load capacitors I am showing here.

Normally for a cMOS circuit every line will be driving 2 transistors 1 pMOS, and 1 nMOS like a here is driving 2 transistors. So, this is the equivalent capacitors. Now let us see; think of a fault in transistor T 1 stuck open see for stuck open, when I am intuitively speaking how I should try and test it I should test it in this way, that this transistor is open let me see if a current can flow here. So, the only test vector I can apply for that is 0 0, because 0 0 is a test vector which makes both T 1 and T 2 conducting and V DD should be connected to the output, that if there is a fault V DD should not get connected, right. So, I am accepting this.

But let us see what will happen here. So, if suppose I apply 0 0. So, in presence of the fault coming is let us say first in the absence of the fault, the output will be V DD or high because both the transistors will conducting, and both the pulled down or off 0 0 means both are off; so the output node is connected to V DD. So, the output is high, which means the capacitor is charging to its full voltage V DD minus this drop. But suppose I have the faulty case where this transistor is stuck open, then if I apply 0 0, I suddenly have a situation where my pull up network is not conducting, because one of them is off. So, F is not connecting to V DD neither F is connected to ground.

So, the output node is electrically disconnected it is like a floating wire, it is just floating. So, there is no path for this capacitor to get charged or discharged. So, the this capacitor will be retaining, the last charge that it was said with the last value of F whether it was low or high low means it was discharged high means it was charged. So, it will remain in that capacitive stage and the output value will show that same thing, the output is floating and the voltage will depend on the charge stored in the load capacitor. So, to (Refer Time: 08:59) this problem what we do, we additionally apply another vector in the beginning 1 0, the target is to set the output at a known state the reverse of this I want to discharge the capacitor. So, if I apply 1 0 then my pulled down network will be on, the capacitor will get discharged by this path output will be 0. So, now, if I apply 0 0, so in the faulty case output will remain 0, in the fall free case output will be 1; I can distinguish, right.

(Refer Slide Time: 09:47)

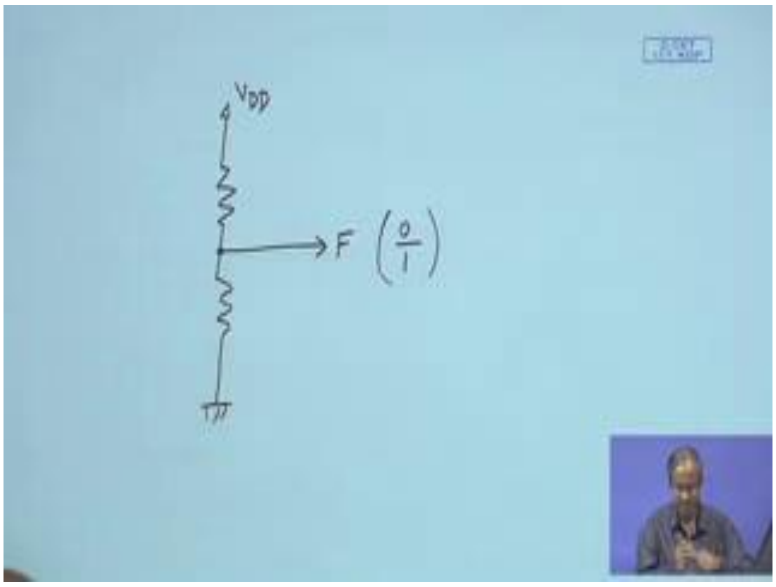


(b) Stuck-short Fault Model

- A transistor is permanently conducting in the presence of a fault.
- Checking the logic value at the output may not be sufficient.
  - Both pull-up (pMOS) and pull-down (nMOS) networks may become conducting, thereby causing the output to reach some indeterminate level.
  - High current will be flowing from V_{DD} to *GND*.
- To detect this type of fault, we need to monitor the current flowing.
  - Called I_{DDQ} *testing*.
  - Test vector causes a conducting path from V_{DD} to *GND* in the presence of fault.

So, I need a pair of patterns to be applied in a particular sequence, this is the characteristic right. Now let us come to stuck short fault, stuck short is the other way round where a transistor is permanently conducting. Now we shall see again through an example therefore, this case just checking the logic value in the output may not be sufficient, because both the pull up and pull down network may be simultaneously conducting. So, you can have a situation where it effectively becomes like a voltage divider, I said becomes like a voltage divider, so here V DD here, this is your pull up conducting resistance pull down conducting resistance and it taking the output from here.

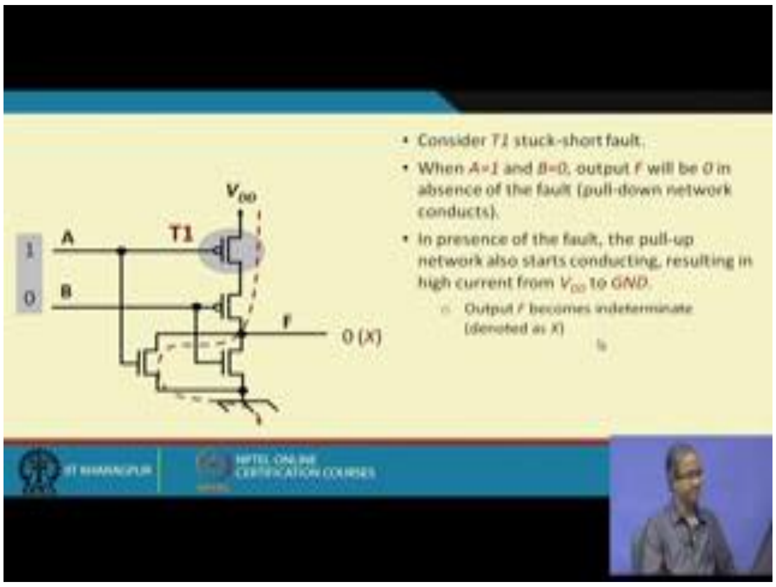
(Refer Slide Time: 10:19)



So, this output may be at a voltage which is between 0 and 1 V DD and ground, which may be neither 0 nor 1 it is somewhere in between. So, just by looking at the logic value you really cannot distinguish right. So, this is the scenario here, because of this fault both the pull up and pull down networks may become conducting. So, the output may reach some in determinant level between 0 and 1. But one thing is true here; here because both the pull up and pull down networks are conducting there will be high current that will be flowing from V DD to ground.

So, to detect this kind of fault, we just cannot look at the output voltage level rather we monitor the current this is sometimes called I DDQ testing, we measure the current flowing from V DD to ground. So, the test factor will be such that it will cause a conducting path from V DD to ground if the presence of the fault, like we take an example of a nor gate again.

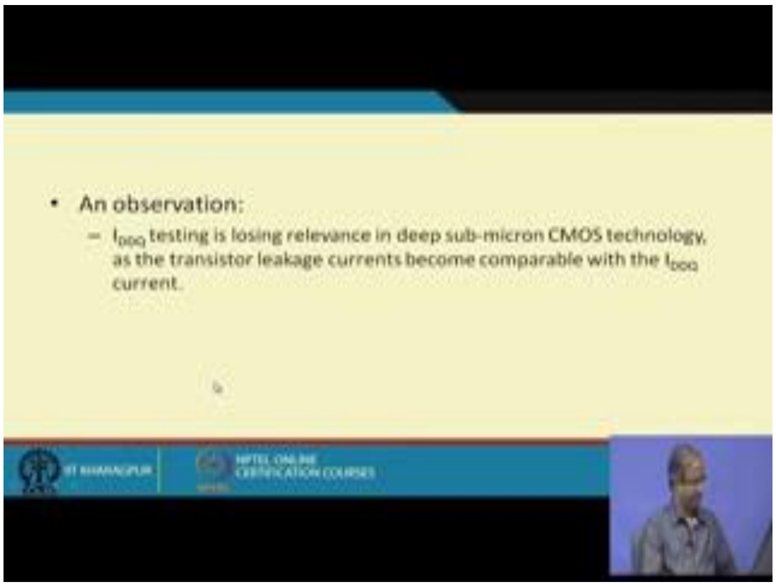
(Refer Slide Time: 11:45)



- Consider $T1$ stuck-short fault.
- When $A=1$ and $B=0$, output F will be 0 in absence of the fault (pull-down network conducts).
- In presence of the fault, the pull-up network also starts conducting, resulting in high current from V_{DD} to GND.
  - Output F becomes indeterminate (denoted as X)

Let us say we take the case of a T 1 stuck short fault, let us say we apply a 1 0 test vector. So, for 1 0 test vector this is supposed to be turned off this is supposed to be on. So, in the fault free case there should not be any path from V DD to F because this will be off, but because of the fault there will be a conducting path, but in the pulled down networks this A is also turning on this transistor. So, what is happening is that there will be a conducting path like this from V DD to ground, there will be a high current flowing because pull up is also conducting, pull down is also conducting, this is what is meant by I DDQ testing we have to measure the current, right.

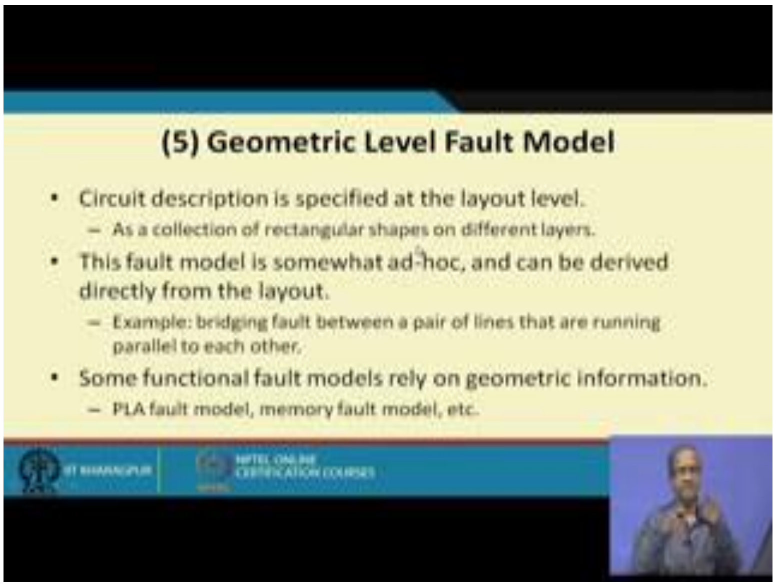
(Refer Slide Time: 12:46)



- An observation:
  - I_{DDQ} testing is losing relevance in deep sub-micron CMOS technology, as the transistor leakage currents become comparable with the I_{DDQ} current.

Now, one observation is that in the present date deep sub micron test technology, this I DDQ testing is becoming difficult to use, because the transistor leakage currents. There are so many transistor billions of transistors in a chip. So, if you take this sum total of all the leakage currents, that becomes comparable with this I DDQ current which you are measuring. So, distinguishing between the short circuit case and the non short circuit case is becoming blurred, the difference in the current will not be that significant right. So, this is one problem.

(Refer Slide Time: 13:29)

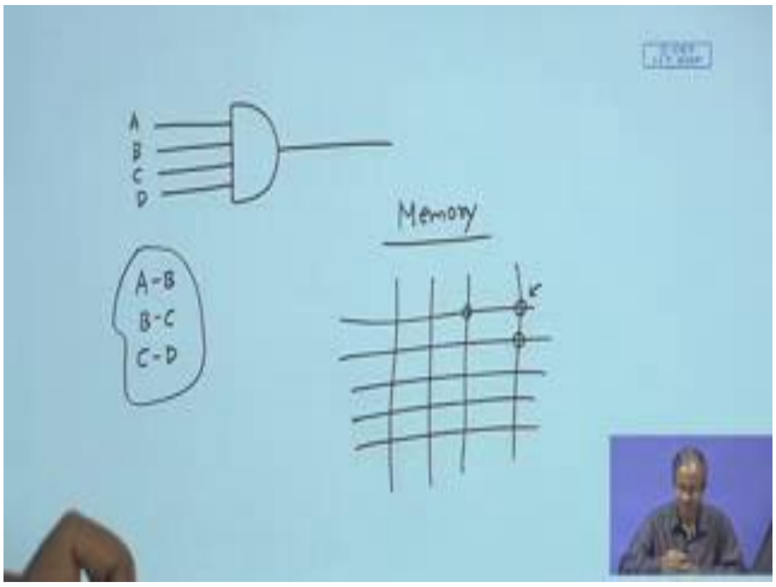


(5) Geometric Level Fault Model

- Circuit description is specified at the layout level.
  - As a collection of rectangular shapes on different layers.
- This fault model is somewhat ad-hoc, and can be derived directly from the layout.
  - Example: bridging fault between a pair of lines that are running parallel to each other.
- Some functional fault models rely on geometric information.
  - PLA fault model, memory fault model, etc.

So, the geometric level fault model we are not going to detail, here we are looking at the circuit description at the layout level. So, we regard the layout level as a collection of rectangular shapes like see if see means we have some information the layout, so we also know that which of the lines have more lightly to be having bridging force like say.

(Refer Slide Time: 13:56)

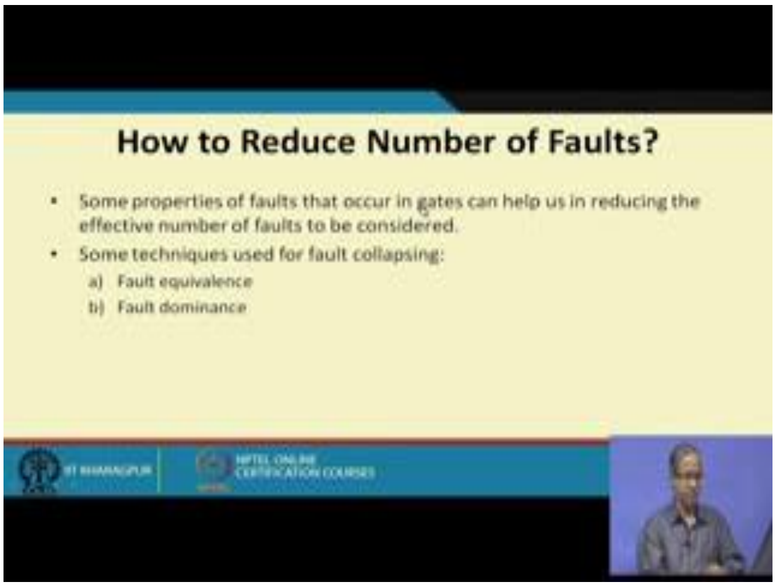


Let us say I have a gate there is a 4 inputs, but in terms of the layout I see that the words are laid out in this order only. So, the chance of bridging fault are let say if A B C and D are there. So, there can be a bridging between A and B between B and C between C and D. At the chance of A and C, or B and D, or A or D having a bridging will be much less because there is a wiring between.

So, if we have some geometric information, you will be knowing that what kind of faults are more likely. Similarly in memory of course, here we shall not be considering memory testing, but let me tell you in memory this cells are arranged in rows and columns, very dense rows and columns. So, if we know which memory cells are near to each other there is something called coupling fault, if you write something into a memory cell the neighbouring cell might get affected.

So, you know which are the neighbouring cells, so you write something here and check whether the neighboring cells are getting effected not the others right, these are all examples of geometric faults models. So, this fault models can be derived directly from the layout.

(Refer Slide Time: 15:35)



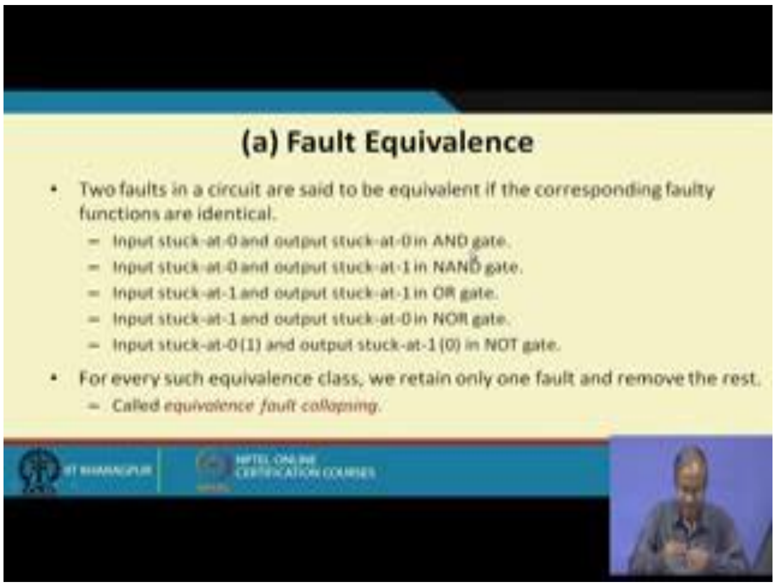
How to Reduce Number of Faults?

- Some properties of faults that occur in gates can help us in reducing the effective number of faults to be considered.
- Some techniques used for fault collapsing:
  a) Fault equivalence
  b) Fault dominance

So, as I said the memory fault models and also fault models per programmable logic arrays we have some regular structures, these are all examples of this. Now let us look into a very important problem, now we had looked at the stuck at fault model and we have said that stuck at fault is the most widely used in particular the single stuck at fault.

Now, for a circuit with let us say 10000 lines, the number of single stuck at fault will be 20000 2 k, but still twenty thousand is a pretty large number of. So, now, we see how we can reduce this number without sacrificing the quality of test. So, here we look at some properties of faults with respect to the gates of course, that can help us in reducing the effective number of faults, this process is called fault collapsing like some of the faults you can collapse into a single fault, we shall be looking at 2 different techniques or approaches fault equivalence and fault dominance.

(Refer Slide Time: 16:40)

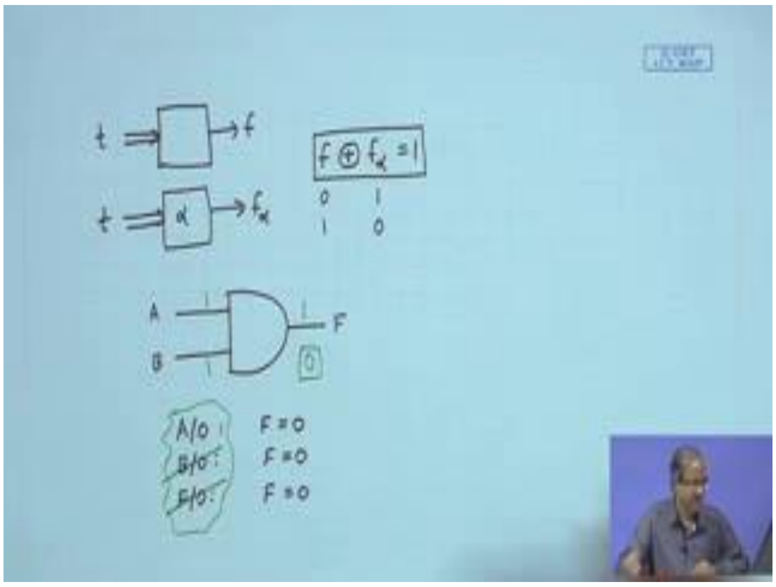


(a) Fault Equivalence

- Two faults in a circuit are said to be equivalent if the corresponding faulty functions are identical.
  - Input stuck-at-0 and output stuck-at-0 in AND gate.
  - Input stuck-at-0 and output stuck-at-1 in NAND gate.
  - Input stuck-at-1 and output stuck-at-1 in OR gate.
  - Input stuck-at-1 and output stuck-at-0 in NOR gate.
  - Input stuck-at-0 (1) and output stuck-at-1 (0) in NOT gate.
- For every such equivalence class, we retain only one fault and remove the rest.
  - Called *equivalence fault collapsing.*

Let us see fault equivalence first. The definition is very simple, 2 faults in a circuit are said to be equivalent if the corresponding faulty functions are identical. Let the first example says the input stuck at 1 and output stuck at 0 and output stuck at 0 in and gate.

(Refer Slide Time: 17:06)



Let us try to explain, you see suppose I have a function any block the output is f, let us say there is a fault in this function the fault is alpha. So, in presence of fault the output is f alpha. So, I have applied some test input let us say t, let us say I applied to the good function as well as the faulty function.

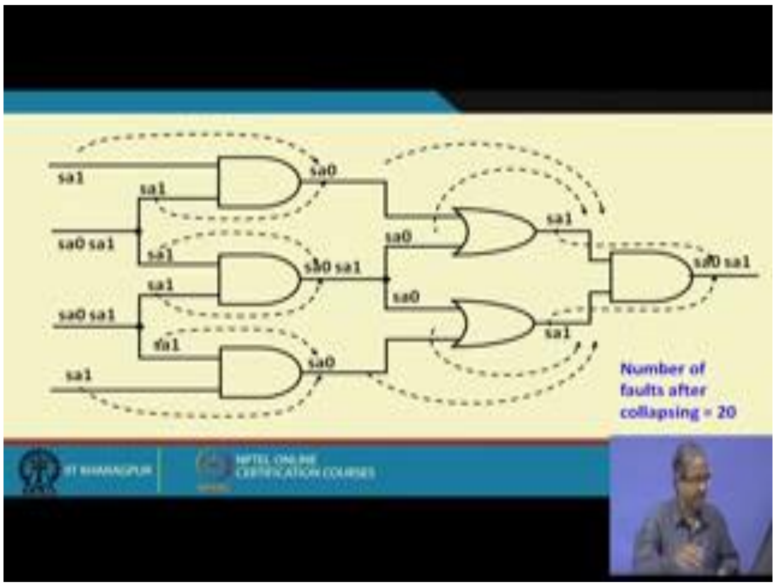
So, if I see that the XOR of f and f alpha equal to 1, which means it is either 0 1 or 1 0 which means they are different then only we say that the fault is getting detected right? Now think of an AND gate, let us take a 2 input and gate let say A B F. So, if I have a fault A stuck at 0 here output F will be 0, because one of the input of the and gate is 0 if I have a fault B equal to 0 then also output F equal to 0 . So, if I have a fault F stuck at 0 then also F equal to 0.

So, I can detect these faults by applying clearly a text vector 1 1; because here my fault free output F is supposed to be 1 and F alpha is 0 their XOR is 1. At now tell me one thing in presence of the fault the output will be 0, now if I see that the output is 0. So, output is coming to 0. So, can I distinguish whether it says A stuck at 0, B stuck at 0 or F stuck at 0? I cannot distinguish; because under the each of these three faults the function changes in a same way it is becoming 0, a constant 0 function.

So, just by looking at this output 0 I cannot distinguish between them. So, what I am saying is that do I need to keep all this 3 faults? So, what if I delete any 2 of them and I keep only a stuck at 0. So, if are detecting a stuck at 0 anyway I have to apply 1 1, you see this test vector will also be detecting b stuck at 0 f stuck at 0. So, I need not keep all 3, I will remove 2 and I will keep only 1 right this is called a equivalence class, that is what I have shown in the slides. So, the first equivalent class is that input stuck at 0 and output stuck at 0 and gate, but if it is a nand gate that input stuck at 0 and output stuck at 1 will be equivalent, they are all equivalent they will all make output equal to 1. For or gate the input and output stuck at 1 falls or equivalent, for a nor gate input stuck at 1 and output stuck at 0 or equivalent, this you can check and for NOR gate there are 2 equivalence classes; Input stuck at 0 is equivalent to output stuck at 1, and input stuck at 1 is equivalent to output stuck at 0.

So, as I said. So, if I can identify every such equivalence class we keep only 1 fault from that class and remove the rest, this process is called equivalence fault collapsing like in the example I have shown here we have kept only 1, we have removed the other 2 this process is called equivalent fault collapsing.

(Refer Slide Time: 20:55)

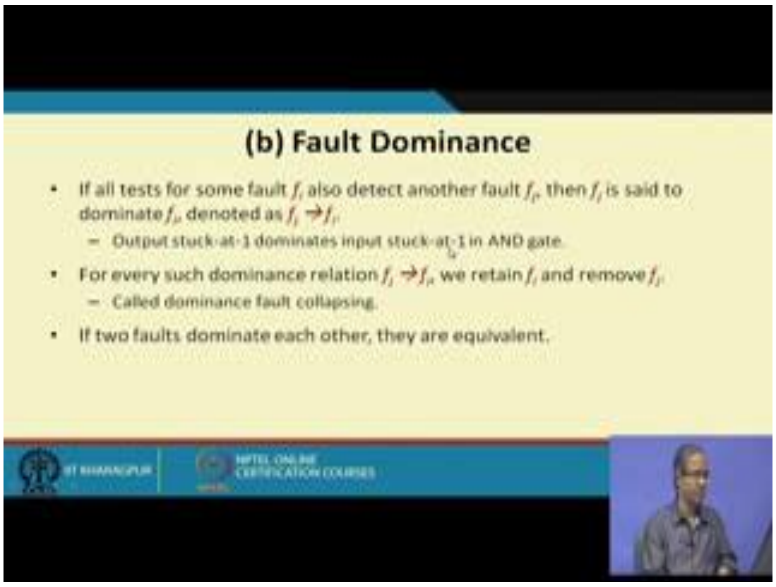


Let us take an example. So, here I have shown a circuit if we count there are 16 lines, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16. So, there are 32 possible signal stuck at fault. So, I am showed them stuck at 0 stuck at 1 and all of them, what I do using this dotted lines I have shown the equivalent faults, like for this and gate the output stuck at 0 is equivalent to input stuck at 0 input stuck at 0 these 3 are equivalent.

Similarly, for this and gate also stuck at 0, stuck at 0, stuck at 0 this and gate also same this or gate stuck at 1, stuck at 1 and the stuck at 1 these are all equivalent this or gate also stuck at 1, stuck at 1, and the stuck at 1 for this and gate stuck at 0, stuck at 0, stuck at 0. So, what I am saying is that for each of this equivalent classes I shall be removing c these are all 2 inputs gates. So, each class is continuing 3 faults for example, for this and gate I shall be removing 2 of them marked as red, I will be keeping only one.

So, I will be repeating for all of the gates, if I do it I get something like this, this was the original thing the red mark faults I have removed by collapsing, I keep the rest. So, you see that after I have done equivalence collapsing the number of remaining faults is only 20. So, early it was 32, now it has reduced to 20. So, this is a process which is typically done before you actually do testing of the circuit or test generation or whatever, right.

(Refer Slide Time: 22:46)



(b) Fault Dominance

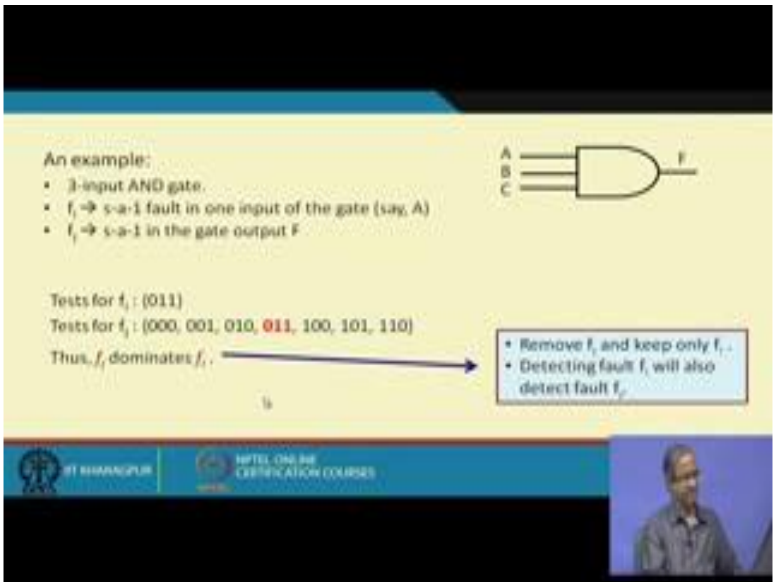
- If all tests for some fault f_i also detect another fault f_j, then f_j is said to dominate f_i, denoted as $f_j \rightarrow f_i$.
  - Output stuck-at-1 dominates input stuck-at-1 in AND gate.
- For every such dominance relation $f_j \rightarrow f_i$, we retain f_i and remove f_j.
  - Called dominance fault collapsing.
- If two faults dominate each other, they are equivalent.

IIT KHARAGPUR
NPTEL ONLINE
CERTIFICATION COURSES

There is another method using which also we can reduce the number of faults this is called fault dominance.

Fault dominance is a little different, the definition goes like this it says if all tests for some fault f I also detect another fault f j, then f j set to dominate f i we denote it like this. We are saying if all tests for some fault f i detect another fault f j then f j dominates f i and if such a property is there what I can do we can remove f j; we keep only f i, because we know that test for f i will automatically detect f j that is a definition. So, if you only keep f i and generated test for f i that will guarantee that f j will also get detected, and if it is a mutual relationship f j dominates f i and f i dominates f j they are equivalent.

(Refer Slide Time: 24:09)



An example:

- 3-input AND gate.
- $f_i \rightarrow$ s-a-1 fault in one input of the gate (say, A)
- $f_j \rightarrow$ s-a-1 in the gate output F

Tests for f_i : (011)

Tests for f_j : (000, 001, 010, 011, 100, 101, 110)

Thus, f_j dominates f_i .

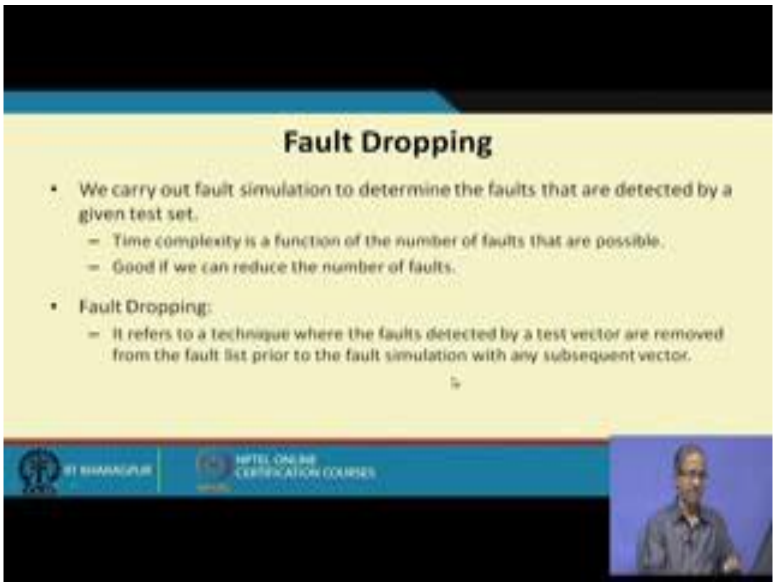
- Remove f_j and keep only f_i .
- Detecting fault f_i will also detect fault f_j.

Now, let us take an example let an AND gate, the output stuck at 1 fault dominates input stuck at 1 fault let us see how. Take a 3 input and gate. So, we define f i as stuck at 1 fault in let us say line A and f j as stuck at 1 fault in f. So, for detecting stuck at 1 fault in line A what are the possible test vectors? There is only 1 test possible 0 1 1, because if B and C are anything other than 1 1 output will be permanently 0.

So, I want to apply something where B and C will be allowing the fault effect to propagate; that means, 1 1. These are the so called non controlling values that we had discussed earlier if you recall, and A is the point where fault is their; if there is a fault this will be 1, if there is no fault I have applied the 0 . So, fault free case output is 0 faulty case output is 1 there is a single test, but for f j which is stuck at 1 in the output, so any test which makes output 0 will be a test for this fault, because I am trying to check whether the output is stuck at 1 or not. So, I can apply any one this 7 vectors, they all make output 0 , but if there is a fault output will be 1 different, right.

Now, here we see that the test for f i is a proper subset of the test for f j. So, I can remove f j from my list, because I know if I detect f I which I can do by this vector, this vector is also included in the test set for f j. So, these are example or f j dominates f i and I can remove f j remove f j and keep f I detecting f i will also guarantee the detection of f j fine.

(Refer Slide Time: 26:18)



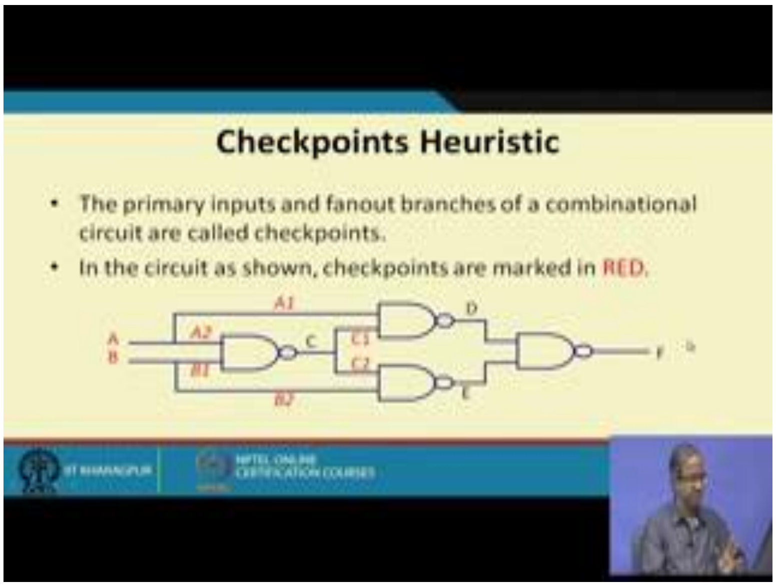
Fault Dropping

- We carry out fault simulation to determine the faults that are detected by a given test set.
  - Time complexity is a function of the number of faults that are possible.
  - Good if we can reduce the number of faults.
- Fault Dropping:
  - It refers to a technique where the faults detected by a test vector are removed from the fault list prior to the fault simulation with any subsequent vector.

So, fault dropping is a process where we use fault simulation is a process we shall be discussing next. So, in faults simulation we detect or determine the set of faults that are detected by given test set.

Now, the complexity if fault simulation depends on the total number of faults. So, if you can reduce the number of faults, fault simulation can run faster. Fault dropping what it says is that suppose I have generated a test for a particular fault, I carry out fault simulation to find out what others faults are also getting detected. So, if I find out the there are 5 other faults which are getting detected, so let me remove those 5 faults from the fault list. So, the size of my fault list will get reduced, this is called fault dropping. It refers to a technique where the faults detected by test vector are removed, prior to the fault simulation with any subsequent vector.

(Refer Slide Time: 27:31)



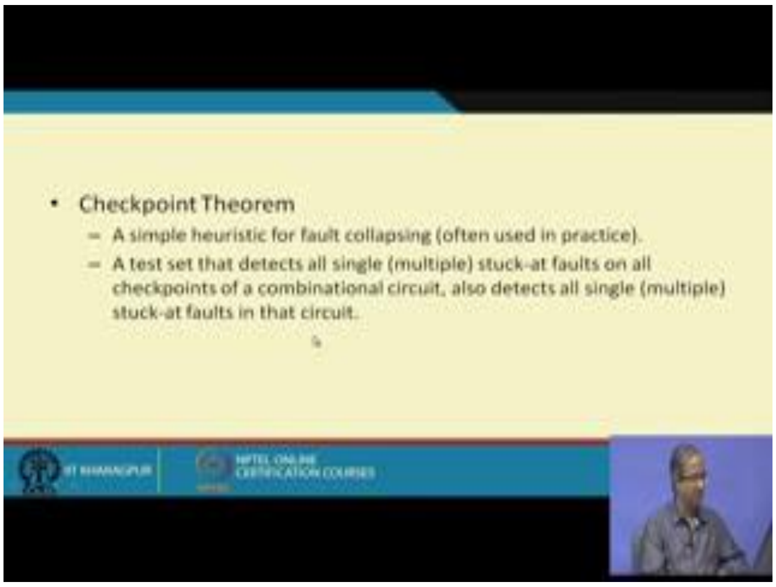
Checkpoints Heuristic

- The primary inputs and fanout branches of a combinational circuit are called checkpoints.
- In the circuit as shown, checkpoints are marked in RED.

A1
A2
A
B
B1
B2
C
C1
C2
D
E
F
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, there is a heuristic which is available which simplifies out process of identifying the equivalent and dominant faults, there is the heuristic called checkpoints. Now checkpoints is defined as for a circuit, the primary inputs and the fanout branches. Like in this circuit the 2 input XOR function, the primary inputs are A B, and the fanout branches A 1 2, B 1 B 2, C 1 C 2 these are the checkpoints C D E F are not. So, there is a checkpoint theorem which says that a test set that detects all stuck at faults on the checkpoints, also detects stuck at faults in this all stuck at faults in this circuit.

(Refer Slide Time: 28:05)



- Checkpoint Theorem
  - A simple heuristic for fault collapsing (often used in practice).
  - A test set that detects all single (multiple) stuck-at faults on all checkpoints of a combinational circuit, also detects all single (multiple) stuck-at faults in that circuit.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, the motivation is that you can use the checkpoint theorem as a simple heuristic fault collapsing means, in this circuit we identify the checkpoints just keep the checkpoints and delete all the fault and all other lines, because you know that fault and all others lines will be either equivalent to some other fault or they will get dominated by some other faults. So, checkpoint heuristic is a very quick method to do this fault collapsing right and this is often used in practice.

So, with this we come to the end of this lecture, now in our next lecture we shall be looking at the problem of fault simulation.

Thank you.