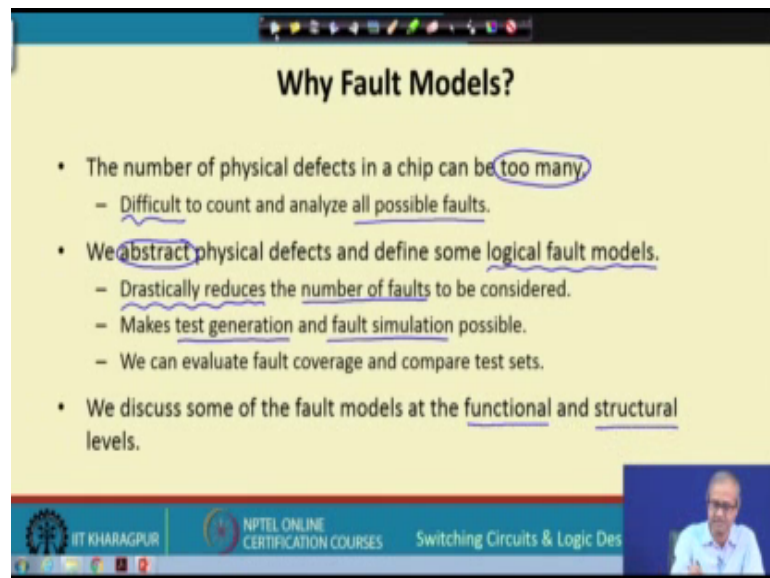


**Switching Circuits and Logic Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 56**  
**Fault Modeling**

In this lecture, we shall be starting the discussion on Fault Modeling. So, if you recall, we discussed in the last lecture that fault modeling is one very important step that we should do at the very beginning before we go through the other steps like test generation, fault simulation etcetera ok. So, the topic of this lecture is fault modelling.

(Refer Slide Time: 00:47)



**Why Fault Models?**

- The number of physical defects in a chip can be too many.
  - Difficult to count and analyze all possible faults.
- We abstract physical defects and define some logical fault models.
  - Drastically reduces the number of faults to be considered.
  - Makes test generation and fault simulation possible.
  - We can evaluate fault coverage and compare test sets.
- We discuss some of the fault models at the functional and structural levels.

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

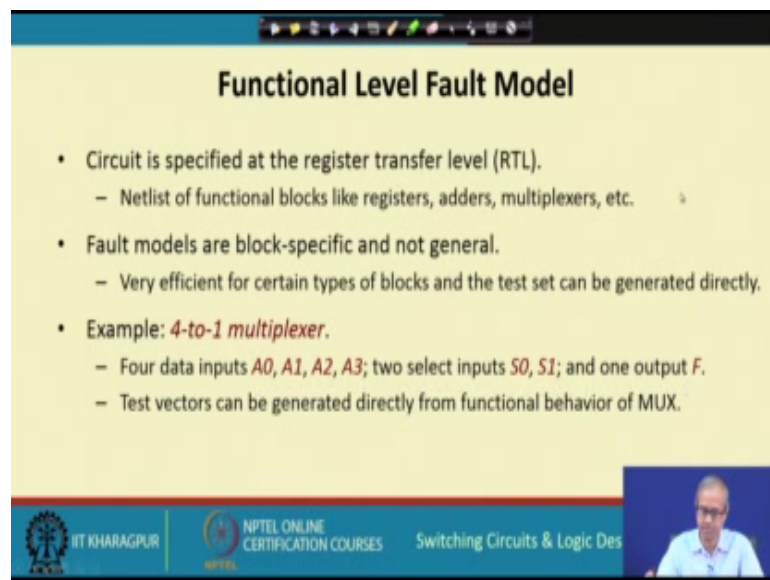
So, let us start with the basic motivation once more that why we need fault models. So, we mentioned in the last lecture that the number of physical defects in a chip can be too many. So, you really cannot count how many physical defects can be there in a chip, because most of the parameters are continuous in nature, and there can be infinitely possible values. So, you cannot test the circuit against all possible values of the parameters ok.

So, it is difficult I should not say difficult, it is impossible to count and analyze all possible faults, so because of this we do some kind of an abstraction. We abstract the defects and define some so called logical fault models. So, if we do this, the advantage that we get is the number of faults to be considered can be drastically reduced. We shall

see some examples which will support this statement. And if we can reduce the number of faults, the subsequent steps of test generation and fault simulation becomes that much easier ok. And if we can count the number of faults, then we can also compute the value of fault coverage.

We can analyze a circuit with respect to testability that how many of faults are getting detected, which are the faults which are not so easy to detect and so on ok. So, we shall be discussing a few of the fault models which are very important. There are many other fault models as well just you should remember. So, we shall be discussing some fault models at the functional and structural levels.

(Refer Slide Time: 02:49)



The slide is titled "Functional Level Fault Model" and contains the following content:

- Circuit is specified at the register transfer level (RTL).
  - Netlist of functional blocks like registers, adders, multiplexers, etc.
- Fault models are block-specific and not general.
  - Very efficient for certain types of blocks and the test set can be generated directly.
- Example: *4-to-1 multiplexer*.
  - Four data inputs  $A_0, A_1, A_2, A_3$ ; two select inputs  $S_0, S_1$ ; and one output  $F$ .
  - Test vectors can be generated directly from functional behavior of MUX.

The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and the text "Switching Circuits & Logic Des". A small video inset in the bottom right corner shows a man speaking.

So, we start with a functional level fault model. So, functional level fault model means the circuit that we are trying to test. We have the description of the circuit given at the functional level. What do mean by functional, level functional level is sometimes also called register transfer level or RTL level. Here the basic building blocks of these circuits are not gates, but slightly higher. We talk about registers, adders, multipliers, multiplexers and so on. So, we have a circuit at that level that is something which we called functional level.

And the point to notice that fault models that are defined at the functional level they are very specific to blocks like for a multiplexer I will have some fault model, for an adder I will have a different fault model, they are not general. They will be very specific to the

blocks that are being used, but let us say for a multiplexer. If you have a strategy that will be very efficient, I will take an example for a multiplexer you can see that the test set can also be generated fairly directly this we shall be seeing. So, this 4 to 1 multiplexer I shall be taking as an example, where there are 4 data inputs 2 select inputs and one output.

(Refer Slide Time: 04:37)

• For a  $2^n$ -to-1 MUX,  $2^{n+1}$  test patterns are required.

• All functional faults can be detected.

- An input line is incorrectly selected.
- Some line is fixed at 0 or 1.

$2^6 = 64$

A0	A1	A2	A3	S0	S1	F
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	1	1	1	0	0
0	1	0	0	1	0	1
1	1	0	1	0	1	0
0	0	1	0	0	1	1
1	1	1	0	1	1	0
0	0	0	1	1	1	1

So, let us take this example. So, you look at the multiplexer schematic diagram here. The 4 inputs are here a 0 to A 3, select lines S 0 and S 1 and the output is F. The corresponding truth table is shown here. Well, here the assumption is that S 0 S 1 if it is 0 0, then we are selecting A 0 ok. 0 0 means A 0 is getting selected. If it is a 0 1 0 1 that means S 1 is 0 S 0 is 1, S 0 is the least significant bit 0 1, then A 1 is selected. If it is 1 0, S 1 is 1 S 0 is 0, then A 2 is selected. And if it is 1 1 then A 3 is selected. So, with respect to the test generation what we do, you see we apply the 4 possible values of the select lines 0 0, 0 1, 1 0 and 1 1

Now, we know when I apply 0 0, this A 0 is supposed to be selected. So, we apply both condition if A 0 is 0, then the output is supposed to be 0. If A 0 is 1 output is supposed to be 1 ok. But we also do something more, like we are talking about a functional fault model, like we know how a multiplexer works ok. If S 0 S 1 is 0 0 A 0 is supposed to be selected, but because of a fault at the functional level what you may say that well, because of some fault instead of A 0 maybe some other input is getting selected maybe A 1 maybe A 2 maybe A 3. So, just to safeguard against that what I do you see, when A 0 is

0 we are applying reverse values in A 1 A 2 or A 3, so that if accidentally any other of them are getting selected, so the output will not be 0 it will become 1 and it can be detected.

Similarly, for A 0 when it is 1 then we are applying reverse values in the others, same thing for the other combinations for 0 1. This A 1 is made 1 1s and 1 and the other inputs are at the reverse value 1 1 1 or 0 0 0, for 1 0 A 2 same and for 1 1 A 3. So, you see at the multiplexer just looking at how a multiplexer works functional level we require eight test vectors only, but if you look at the total number of inputs, there were 6 with respect to the truth table  $2$  to the power  $6$  is  $64$ . So, instead of applying  $64$  possible inputs we need to apply only  $8$  inputs to test the functional behaviour of a multiplexer right.

This is the basic idea. We need not have to apply all possible inputs. We have to understand the functional behaviour of the circuit. We want to test with respect to that we can directly generate the test vectors fine. So, in general for a  $2$  to the power  $n$  to  $1$  line multiplexer with  $n$  select lines, we need  $2$  to the power  $n$  plus  $1$  test vectors. Here this  $n$  was  $2$  that is why we need  $2$  the power  $3$  equal to  $8$  test vectors.

And in this scheme all the functional faults can be detected, like what are the function faults, we are talking about that some input line is incorrectly selected as I had said or you may check some of the line they are permanently fixed at  $0$  or  $1$ . So whatever you are applying from outside instead of that value it is always at  $0$  or always at  $1$ . Some line maybe accidentally short circuited or you are connecting it is not connected it will treat as  $0$  maybe, because of those kind of failures this kind of faults can occur.

So, this example shows that functional fault model can be very effective. And once you have a functional fault model you can have a very efficient web generating the tests, but again whatever we talked about for a multiplexer you cannot apply it for an adder, for an adder the logic and the behaviour will be entirely different. So, these techniques are very specific to blocks alright. So, these are not general.

(Refer Slide Time: 09:45)

**Structural Level Fault Model**

- Circuit is specified as a netlist, typically at the level of gates and flip-flops.
- The assumption:
  - The blocks (e.g. gates) are fault-free.
  - The interconnections between blocks can be faulty.
- We discuss one popular structural level fault model, viz. *stuck-at fault model*.

The slide includes a hand-drawn circuit diagram showing two AND gates connected to an OR gate. The bottom right corner features a small video inset of a speaker.

NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Des

So, let us now look at structural level fault models which are more common. So, structural level means, we view this circuit as a netlist. Netlist means: some basic blocks and their interconnections typically the blocks are considered as gates or flip-flops, for combinational circuits it will be only gates for sequential circuits. There will also be flip flops and their interconnection. Let us say I have a circuit like this. So, this structural fault model that we are talking about it rests on a very important assumption.

The assumption says that the gates or the blocks they are fault free. There cannot be any fault inside the blocks rather faults can be there in the interconnection lines. So, all the interconnection lines there can be faults only there, but you will argue that why this kind of a assumption, there can be fault inside the gate also. But, the logic is there can, of course be a fault inside a gate, but you think the fault may be such that because of that one of the inputs is permanently being treated at, as if it is at 0.

So, you abstracted in a way that the gate you are assuming to be fault free, but that input you are saying that it is permanently at 0. This is the kind of fault you are talking about. Now although, it seems to be overly simplistic, but this has been found to be very effective. And it can cover a very large percentage of the actual physical faults right.

(Refer Slide Time: 11:51)

**Structural Level Fault Model**

- Circuit is specified as a netlist, typically at the level of gates and flip-flops.
- The assumption:
  - The blocks (e.g. gates) are fault-free.
  - The interconnections between blocks can be faulty.
- We discuss one popular structural level fault model, viz. stuck-at fault model.

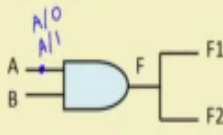
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Des

Now, we shall be discussing one very popular structural level fault model which is called the stuck-at fault model, which is widely used in the industry also.

(Refer Slide Time: 12:07)

**Stuck-at Fault Model**

- Here we assume that some of the circuit lines are permanently fixed at logic-0 or logic-1 due to some failures.
- Very popular fault model, as it can model many realistic physical failures.
  - Faults on a line  $A$  are denoted as:  $A$  s-a-0 or  $A/0$  and  $A$  s-a-1 or  $A/1$
  - Fanout stems and fanout branches are considered as separate lines.



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

Let us think see what is the stuck-at fault model. In the stuck-at fault model the basic assumption is like this we assume that, because of the failures or faults some of the circuit lines are permanently fixed at either logic 1 or logic 0. And we call them as stuck-at 0 or stuck-at-1 fault. The faults are denoted like this A stuck-at 0 s a 0 or in short we can also write like this A slash 0 or a stuck-at-1 or in short I can write A slash 1 ok. So,

suppose this is one line of the circuit I can say that there can be a fault a stuck-at 0, there can be a fault a stuck-at-1 alright.

(Refer Slide Time: 13:11)

**Stuck-at Fault Model**

- Here we assume that some of the circuit lines are permanently fixed at logic-0 or logic-1 due to some failures.
- Very popular fault model, as it can model many realistic physical failures.
  - Faults on a line  $A$  are denoted as:  $A$  s-a-0 or  $A/0$ , and  $A$  s-a-1 or  $A/1$ .
  - Fanout stems and fanout branches are considered as separate lines.

The diagram shows an AND gate with inputs A and B and output F. A checkmark is next to the gate. Two lines, F1 and F2, branch off from the output F. F1 is connected to an OR gate with input F1/0 and a checkmark. F2 is connected to another AND gate. A small video inset of a speaker is in the bottom right corner.

Now, there is another important point also fanout stems and fanout branches are considered as separate lines. Like you consider this F F 1, F 2 this is a fanout stem. And these are the fanout branches. Let us try to understand why this is so you see this F 1 can be connected to some other gate, this F 2 can be connected to some other gate right. Now, you recall that in this fault model we are assuming that these gates are free of faults. Faults can only be there in the interconnections.

Now, suppose there is a fault in this first gate, because of which the output is always at 0 like F stuck-at 0. Now, this fault will equally affect F 1 and F 2, because the value of F is going to both F 1 and F 2. But imagine that there is a fault in this gate, because of which this line let us call it F 1, this F 1 is stuck-at 0. But this fault behaviour will exhibit only on this is not an electrical fault, but because of a fault this gate input is being treated as if it is always at 0, but F or F 2 will still work correctly. There will be no stuck-at 0 fault in F or F 2, because of this when you count the number of lines fanout stems and fanout branches are considered as separate lines, just remember this ok.



(Refer Slide Time: 14:57)

• **Single stuck-at fault**

- Only one line of the circuit has a stuck-at fault at any given time.
- Most widely used fault model in the industry.
- For a circuit with  $k$  lines, total number of single stuck-at faults is  $2k$ .

$k=12 \rightarrow$  Number of single stuck-at faults = 24

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Des

First let us talk about single stuck-at fault which is the most widely used stuck-at fault model. This assumes that only one line of the circuit has a stuck-at fault at any given time. If a circuit has  $k$  number of lines, so each of the line can have 2 faults stuck-at 0 or stuck-at-1, because only one of them is occurring at a time total number of faults will be  $2k$ . Let us, take a gate level netlist here. This is the NAND realization of the XOR function. You see here there is a fanout, and here also there is a fanout. You count the number of lines 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12, so  $k$  is 12, because  $k$  is 12 total number of stuck-at single stuck-at faults can be  $2k$  or 24 simple.

(Refer Slide Time: 16:21)

• **Multiple stuck-at fault**

- Any number of circuit lines can have stuck-at faults at any given time.
- For a circuit with  $k$  lines, total number of multiple stuck-at faults is  $3^k - 1$ .

$k=12 \rightarrow$  Number of multiple stuck-at faults =  $3^{12} - 1 = 5,31,440$

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Design



Now, people have also talked about multiple stuck-at faults which mean more than one lines having stuck-at faults at the same time. So, here something is mentioned that for a circuit with k lines total number of multiple stuck-at fault can be 3 to the power k minus 1, but how it is coming 3 to the power k minus 1. You just imagine like this; this circuit has k number of lines; think of each of the individual lines one at a time; say the first line it can be in 3 different states; it can be stuck-at-1; it can be stuck-at 0 or it can be go to fault free.

The second line can also be in one of 3 states stuck-at 0 stuck-at-1 fault free. So, all k lines can be in three states each. So, if you look at all possible combinations, it will be 3 multiplied by 3 multiplied by 3 k times 3 to the power k ok. And out of this 3 to the power k there will be one combination which says all of them are fault free. So, you subtract that 1, so it becomes 3 to the power k minus 1 right.

So, for the same example there are twelve lines you see multiple stuck-at faults 3 to the power 12 minus 1 becomes about 5.3 lacks. So, just imagine for this very small circuit with 4 gates the number of multiple faults is becoming 4 lacks. So, what about a large gate with 1000 large circuit with 1000 gates or more ok; so counting multiple stuck-at faults may become impractical that is why most of the people they concentrate on single stuck-at faults only fine.

(Refer Slide Time: 18:25)

**Single Stuck-at Fault Testing Examples**

**2-Input AND Gate:**

Logic diagram: Inputs A and B, Output F. Handwritten notes:  $A \rightarrow 0$ ,  $B \rightarrow 0$ ,  $F \rightarrow 0$ ,  $1 \rightarrow 0$ .

Test Set:  $T = \{01, 10, 11\}$

- 01 : detects A/1 and F/1
- 10 : detects B/1 and F/1
- 11 : detects A/0, B/0 and F/0

Handwritten equation:  $F \oplus F_{\alpha} = 1$

**4-Input AND Gate:**

Logic diagram: Inputs A, B, C, D, Output F.

Test Set:  $T = \{0111, 1011, 1101, 1110, 1111\}$

- 0111 : detects A/1 and F/1
- 1011 : detects B/1 and F/1
- 1101 : detects C/1 and F/1
- 1110 : detects D/1 and F/1
- 1111 : detects A/0, B/0, C/0, D/0 and F/0

NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Des

Now, let us look at some examples of single stuck-at fault testing. You think of a single 2 input and gate. So, there can be 6 faults a stuck-at 0 A stuck-at-1, B stuck-at 0, B stuck-at-1, F stuck-at 0 F stuck-at-1. So, I am showing the set of tests that can detect all faults. So, I tell you 0 1, 1 0 and 11 are sufficient. 01 if we apply 01, then under fault free condition output will be 0.

So, if either A stuck-at-1 fault is there, then 1 will come the output will change to 1 it can be detected or output F stuck-at-1 fault is there, then also out become 1 it can be detected. Similarly, 1 0 can detect B stuck-at-1 and F stuck-at-1, and 1 1. If we apply both 1, let us say 1 and 1, so output is supposed to be 1. So, if there is a fault A stuck-at 0 that can be detected, because one of them is becoming 0 output will be coming 0. The second 1 B stuck-at 0 can also be detected that will also change. And third one this F stuck-at 0 can also be detected

So, you can detect a fault if the fault free output, and the fault faulty output let us called  $F_{\alpha}$  they are different, which means they are exclusive odd is equal to 1. Exclusive add equal to 1, when they are either 0 1 or 1 0 means they are different. This is the necessary condition for the detection of a fault ok. Take a larger and gate for input and gate. Following a similar logic you can see this you can verify that you require only 5 test vectors for testing all faults 0 1 1 1. This will detect only a stuck-at-1 and F, stuck-at-1 1 0 1 1 B and F, C and F and this 1 D and F. But 1 1 1 1 will detect all the stuck-at 0 faults. So, this 5 test vectors taken together can detect all the 10 faults right

(Refer Slide Time: 21:03)

• 3-input XOR gate  $T = \{000, 111\}$

- 000 detects all input stuck-at-1 faults and also output stuck-at-1 fault.
- 111 detects all input stuck-at-0 faults and also output stuck-at-0 fault

• 4-input XOR gate  $T = \{0000, 1111, 0111\}$

- 0000 detects all input stuck-at-1 faults and also output stuck-at-1 fault.
- 1111 detects all input stuck-at-0 faults and also output stuck-at-1 fault
- 0111 detects additionally the output stuck-at-0 fault.

• For  $m$ -input XOR gate, number of test vectors required will be either 2 (if  $m$  is odd) or 3 (if  $m$  is even).

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

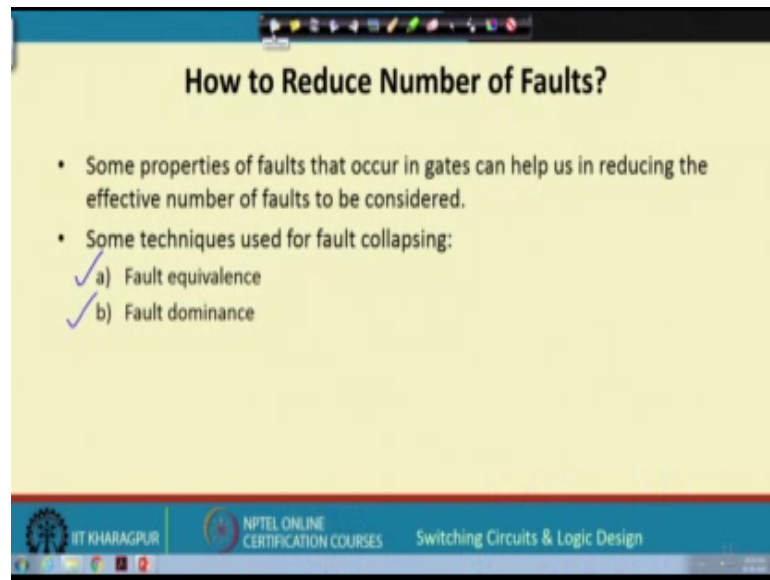
Let us, take some more example let us take a 3 input exclusive or gate 3 input XOR gate. So, I am saying that 2 test vectors are sufficient, why let us say you apply 0 0 0, the output is 0. The property of an exclusive or function is that whenever one of the input changes the output also changes. So, if the first one changes from 0 to 1 that means there is a stuck-at-1 fault, it will be detected. Second line stuck-at-1 fault that will also be detected, third one stuck-at-1 that will be detected, and output stuck-at-1 of course, it will be detected.

Similarly, for 1 1 1 if we apply 1 1 1 in the input, the output will be 1. So, any stuck-at 0 faults will make the output 0. So, this 2 test vectors are sufficient, but for a 4 input XOR gate means 4 means in general for an even input XOR gate this will be true for any odd number, like for an  $m$  input XOR gate. Well, if it is odd you need only 2 test vectors like this all 0s and all 1s, but you have even number of inputs. Then you will be needing 0 0 0 0, but the output is 0 this will detect all stuck-at-1 faults in the input, and also stuck-at-1 in the output. But if we apply 1 1 1 1, for that also even number of 1 that also output will be 0.

This will detect all stuck-at 0 faults in the input stuck-at-1 fault in the output, but only stuck-at 0 fault in the output is missing for that you have to add another test vector. Any test vector with odd number of ones which will make the output one that can detect a stuck-at 0 fault in the output ok. So, for an XOR gate if the number of input is even unit,

3 test vectors. So, these examples illustrate that in the process of test generation number of test vectors can be drastically reduced ok. So, even for a 4 input XOR gate or a 100 input XOR gate you need only 2 or 3 test vectors ok. So, these are very simple examples to illustrate.

(Refer Slide Time: 23:43)



Now, there is one other thing I want to talk about the problem of reducing the number of faults. Like, there are some properties that we shall be talking about 2 properties called fault equivalence and fault dominance that can actually reduce the number of faults we need to consider in a gate ok. So, let us take some examples, it will be clear.

(Refer Slide Time: 24:17)

**(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.

Handwritten diagram of an AND gate with inputs A and B, and output F. A table next to it lists fault conditions: A/0, B/0, and f/0, all resulting in f=0.

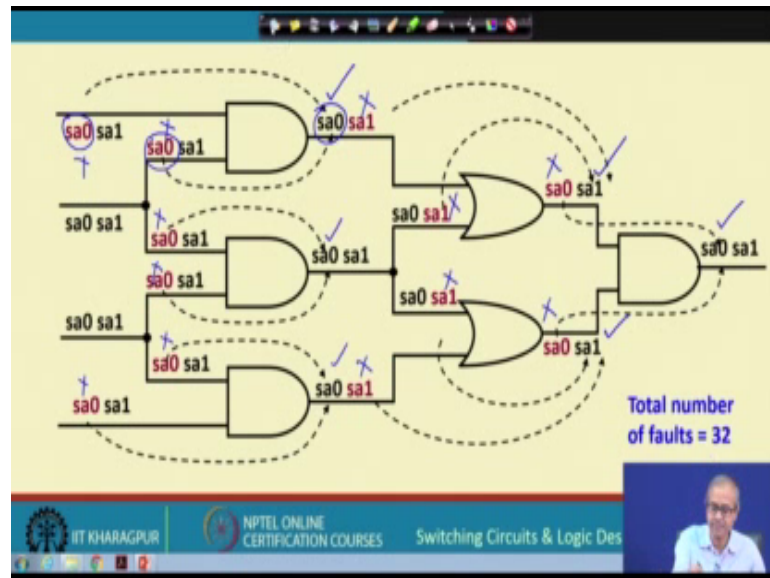
NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Des

First let us talk about fault equivalence. Well I am taking couple of examples others you can verify. Similarly you think of an and gate let us say inputs are A B output is F. So, if there is a fault A stuck-at 0, what will happen so one of the input is always at 0 so the output F is always 0. If there is a stuck-at fault B stuck-at 0, then also output is 0. So, if there is the fault output F stuck-at 0, then also output is 0. So, you cannot distinguish between these 3 faults, they are equivalent ok.

So, the idea is that when you are counting the number of faults, you are counting A stuck-at 0, B stuck-at 0, F stuck-at 0 three times, but you can delete two of them, if you want. What I am saying is that you can delete two of them. You can only keep A stuck-at 0, because these three are equivalent. And you only generate a test vector that can detect a stuck-at 0 fault which is 1 1 output is 1. And this test vector can also detect B stuck-at 0 F stuck-at 0 automatically.

Similar is the case for the other gates you can verify for a NAND gate the input stuck-at 0 and output stuck-at-1 are equivalent, for an OR gate all stuck-at-1 faults are equivalent input stuck-at-1, output stuck-at-1 they will all make the output F equal to 1 like this. Now, this kind of a thing that when there are a set of equivalent faults you keep one delete the others, this is called equivalence fault collapsing.

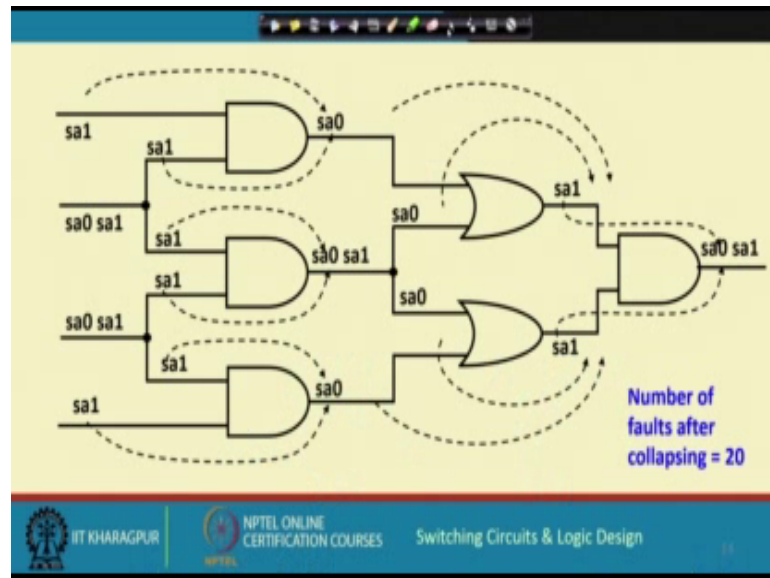
(Refer Slide Time: 26:17)



Let us take an example, let us take a circuit like this. There are some AND gates, and OR gates only we are considering ok. These are fanout stem, so the stuck-at 0 and stuck-at-1 faults will be different. For this AND gate as I have said the stuck-at 0 faults are all equivalent. So, these let us, say these three faults are equivalent stuck-at 0 in the first input stuck-at 0, and stuck-at 0. So, using fault collapsing what I was saying I keep only one other two I delete. The red ones are the ones I have deleted.

Similarly, for the second one I keep this I delete the stuck-at 0, and stuck-at 0. Third one I keep this, I delete this, I delete this. For an OR gate this stuck-at-1 faults are equivalent. So, I keep only one stuck-at fault I delete this, I delete this. Here I keep this I delete this, I delete this. For the last AND gate I keep this, I delete this I delete this. So, how many I have deleted 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

(Refer Slide Time: 27:45)



So, in the original circuit number of faults were 32 ok. And after deletion, so I have remove the red marked ones I am left with 20, this is what is meant by fault collapsing. After fault collapsing using equivalent faults I am able to reduce the number of faults ok. This is one technique.

(Refer Slide Time: 28:05)

### (b) Fault Dominance

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

And the other technique is called fault dominance ok. Fault dominance; let us try to understand the definition then we will take an example. This says if all the tests for some fault let us call the fault is  $f_i$  also detects another fault  $f_j$ , then we define that  $f_i$



dominates  $f_i$  we denote it like that  $f_j \rightarrow f_i$ . So, I have an example I will show like output stuck-at-1 for an AND gate dominates input stuck-at-1, but whenever such a dominance relation is there we will keep a  $f_i$ , and we will be deleting  $f_j$ , we will keep  $f_i$  we will delete  $f_j$ . This is called dominance fault collapsing well. This may look a little confusing let me take an example to explain this will be easiest.

(Refer Slide Time: 29:05)

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  $f_i$ .

Let me take an example, consider a 3 input AND gate ok. And you take the two faults  $f_i$  and  $f_j$  as follows;  $f_i$  is stuck-at-1 fault in one of the input A, let us say A stuck-at-1, this is my  $f_i$ , I call this as  $f_i$ . And  $f_j$ ; let us call the stuck-at-1 in the output F stuck-at-1 this is my  $f_j$ . Now, for this AND gate you see for  $f_i$  what are the tests you can use for testing for a stuck-at-1 there is a single test vector 0 1 1, 0 1 1, because under fault free conditioner F will be 0. If there is a A stuck-at-1, the output will be different, so it will be detected.

But for output F stuck-at-1 fault, there can be many test vectors possible. So, any test which makes the output 0 that will be a test for  $f_j$ , because normally output will be 0 if stuck-at-1 output will become 1. Out of this, you see 0 1 1 is also there. So, what I am saying if such a pair of faults are there, you remove  $f_j$  from the list. Because if you keep  $f_i$ , this 0 1 1 test will this will be included anyway, and 0 1 1 is also a test for  $f_j$ . So,  $f_j$  will also get detected ok. This is the idea behind dominance fault collapsing you keep  $f_i$ , you delete  $f_j$  ok.

(Refer Slide Time: 31:03)

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  $f_j$ .

Diagram: A 3-input AND gate with inputs A, B, and C, and output F.

Footer: IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Des

So, remove  $f_j$  and keep only  $f_i$ , because if you detect this  $f_i$ , it will automatically also detect  $f_j$ .

So, with this we come to the end of this lecture where we have briefly covered the problem of fault modelling. And we have looked at some of the very important and commonly used fault models, we looked at one of the functional fault model for multiplexers, then we talked about the stuck-at fault both single and multiple. So, we shall be continuing our discussion in the next lecture where we shall be talking about the problem of test generation.

Thank you.